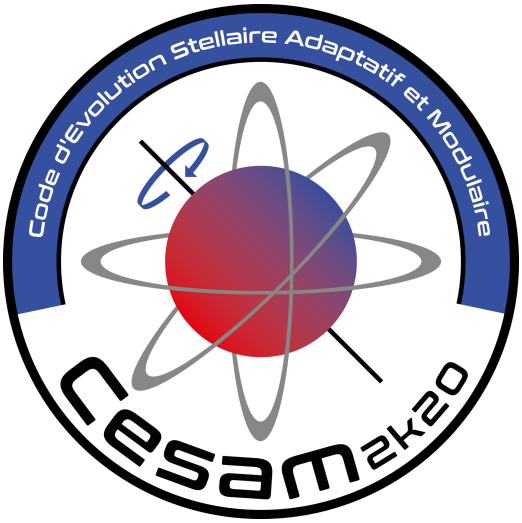
Usage and description of the stellar evolution code Cesam2k20

P. Morel and the Cesam2k20 team

February 29, 2024



Contents

Developers and collaborators

Current developpers

(Developers that you may contact)

Morgan Deal
Yveline Lebreton
Louis Manchon
João Pedro Cadilhe Marques

Past developpers

Pierre Morel*
Georges Alecian
Michel Auvergne
Annie Baglin
Gabrielle Berthomieu
Thierry Corbard
Daniel Cordier
Marie-Jo Goupil
Andy Moya
Laurent Piau
Bernard Pichon
Janine Provost
Réza Samadi

Advisors on specific points

Nathalie Audard
Arnold Boothroyd
Sacha Brun
Roger Cayrel
Jørgen Christensen Dalsgaard
Hervé Dzitko
José Matias
Anthony Noll
Ludovic Petitdemange
Phu Anh Phi Nghiem
Charly Pinçon
Federico Spada
Frédéric Thévenin
Sylvaine Turck-Chièze
Claude van't Veer

---

*: Main historical developer.

Part I

Installation and exploitation

Chapter 1

Cesam2k20

On n'exécute pas tout ce qui se propose
Et le chemin est long du projet à la chose.

Le Tartuffe.

Contents

1.1 Notable versions

The developments of CESAM started around the year 1995. CESAM means Code d'Evolution Stellaire, Adaptatif et Modulaire (Stellar Evolution Codem Adaptative and Modular).

- CESAM2k: With a view to future extensions, a restructuring of the code appeared necessary, mainly for users used to the previous versions of CESAM. It is articulated around a rewriting of the code in FORTRAN95 which allows the grouping of the routines in **module**s. One of the interests of this structure lies in the fact that the compiler checks the concordance of the types (real, integer, etc.), the number and the dimensions of the arguments of the lists of calls to the routines between the calling program and the calling routine. It concerns the following points:
  - dusting off and making the algorithms more reliable,
  - removal of obsolete variables and routines,
  - improvement of the readability, in particular by the use of multi-dimensional tables which, thanks to the modules, are managed more reliably and especially more simply by FORTRAN95 than by FORTRAN77,
  - exploitation in "generic" form of the routines of physics, in order to avoid the trees of externals and thus to facilitate and make reliable the implementation of the extensions and the later modifications,
  - use of programming facilities offered by FORTRAN95 for example: dynamic allocation of arrays, infinite loops, operations on arrays, string management, etc...
  - CESAM2k is an abbreviation of CESAM 2000.
- CESTAM: In the years 2010's, many developments have been made to better model the transport of angular momentum. To differentiate this version from CESAM2k, it was renamed CESTAM, where the 'T' means 'with Transport'.

- Cesam2k20: Since 2018, and more importantly since 2021 for the preparation of PLATO, intense work have been carried to modernize CESTAM, remove bugs, implement new models, and release a public version. This new version is called Cesam2k20. Public repository is found here. We also created a website here.

A general flowchart of the code can be found in 1.1.

1.2 A note for the non French speaker

CESAM was originally developed by P. Morel in Nice observatory and later, mainly by French scientists. Because of that, the majority of the routines module, etc. have French names. Comments are also very often written in French. This English documentation is a step toward an opening to the international community. Nonetheless, it represents a huge amount of work and you may find mistakes or unclear sentences. Since you have access to the source code, feel free to correct them ! If something does not make sense at all and you do not know how to correct it, please report it to `contactcesam2k20@disroot.org`.

1.3 Role of the modules

Cesam2k20 is divided in modules. Until recently, all modules were written in Fortran90 or later, except for `mod_etat` and `mod_opa` that had been kept in Fortran77 in CESAM2k and CESTAM. Only third party module such as `slatec` or `z14xcotrin21` are still Fortran77 (and will stay so!). They now have been modernized to Fortran90 and to ease this process, each routine have been put in a submodule.

- Module `mod_kind`: Gathers the types of variables.
- Module `mod_communicate`: This module is used to write messages and errors.
    - Submodule `submod_error`: Used for error management.
- Module `mod_hdf5_utils`: Gathers some routines that simplify the handling of HDF5 files. Optional.
- Module `slatec`: Numerical library (`https://netlib.org/slatec/`).
- Module `mod_numerique`: Gathers numerical routines.
    - Submodule `submod_num_saha`: Numerical routines specifically dedicated to the SAHA EoS (Baturin et al., 2013).
- Module `mod_donnees`: Gathers most of the quantities that are fixed during the evolution.
- Module `mod_variables`: Gathers most of the variable quantities during the evolution.
- Module `mod_etat`: Gathers routines concerning the equation of state.
    - Submodule `submod_etat_gong`: Gong EoS.
    - Submodule `submod_etat_ceff`: CEFF EoS.
    - Submodule `submod_etat_eff`: EFF EoS.
    - Submodule `submod_etat_mhd`: MHD EoS (Mihalas et al., 1988).
    - Submodule `submod_etat_opal5Z`: Opal 2005 EoS (Rogers & Iglesias, 1992; Iglesias & Rogers, 1996; Rogers & Nayfonov, 2002).
    - Submodule `submod_etat_saha`: Saha EoS.

- Module z14xcotrin21: Module for the routine z14xcotrin21.f90 that allows Z-interpolation over the full available metallicity range in OPAL EoS (https://www.cita.utoronto.ca/~boothroy/kappa.html).
- Module mod_opa: Gathers the routines concerning the calculation of opacity.
  - Submodule submod_opa_compton: Compton opacities.
  - Submodule submod_opa_compton_Poutanen: Compton opacities, revision of Poutanen (2017).
  - Submodule submod_opa_int_zsx: ? opacities.
  - Submodule submod_opa_gong: Gong opacities
  - Submodule submod_opa_yveline_lisse: OPAL opacities with Bézier smoothing.
  - Submodule submod_opa_opalC0: OPAL opacities with $Z < 0.1$ and accounting for modification of the chemical composition due to nucleosynthesis.
  - Submodule submod_opa_opal2: Livermore Type 2 opacity tables.
  - Submodule submod_opa_yveline: OPAL opacities interpolated with Lagrange polynomials.
  - Submodule submod_opa_yveline_daniel: OPAL or Los Alamos TOPS opacity tables.
  - Submodule submod_opa_mono_OP: OP monochromatic opacity tables
  - Submodule submod_opa_houdek9: OPAL 95 opacity tables interpolated with bi-variate splines (module of G. Houdek).
  - Submodule submod_opa_smooth: OPAL opacity tables with smooth interpolation of T and ρ derivatives of opacity. Suitable for non-adiabatic oscillation computation.
  - Submodule submod_opa_kappa_cond: Conductive opacity from Iben (1975, Appendix A).
  - Submodule submod_opa_cond_yveline: Conductive opacity from Alexander Pothekin (1999).
  - Submodule submod_opa_cond_yveline21: Conductive opacity from Alexander Pothekin (2021).
- Module mod_conv: Gathers the routines concerning convection.
- Module mod_nuc: Gathers the routines concerning thermonuclear reactions.
- Module mod_thermo: Gathers the routines concerning thermodynamics computations: determination of limits CZ/RZ, adiabatic gra, etc.dient
- Module mod_atm: Gathers the routines concerning the restitution of the atmosphere.
- Module mod_alecian: Gathers the routines concerning the calculation of radiative accelerations, following the formalism of G. Alécian.
- Module mod_evol: Gathers the routines concerning the temporal evolution of the chemical composition.
  - Submodule submod_evol2d: Submodule for the 2D modeling of rotation.
- Module mod_static: Gathers the routines concerned with the resolution of the quasi-static equilibrium.
- Module mod_cesam: Gathers the routines concerned with the management of the calculation.
- Module mod_exploit : Gathers the routines concerned with the exploitation of the results. This module is located outside /src directory, in /exploit.

A general flowchart of the code is shown in Fig. 1.1. Each routine, private and/or public, is introduced as an **include** in its parent module, with the name of the module indicated in comments at the beginning of each routine.

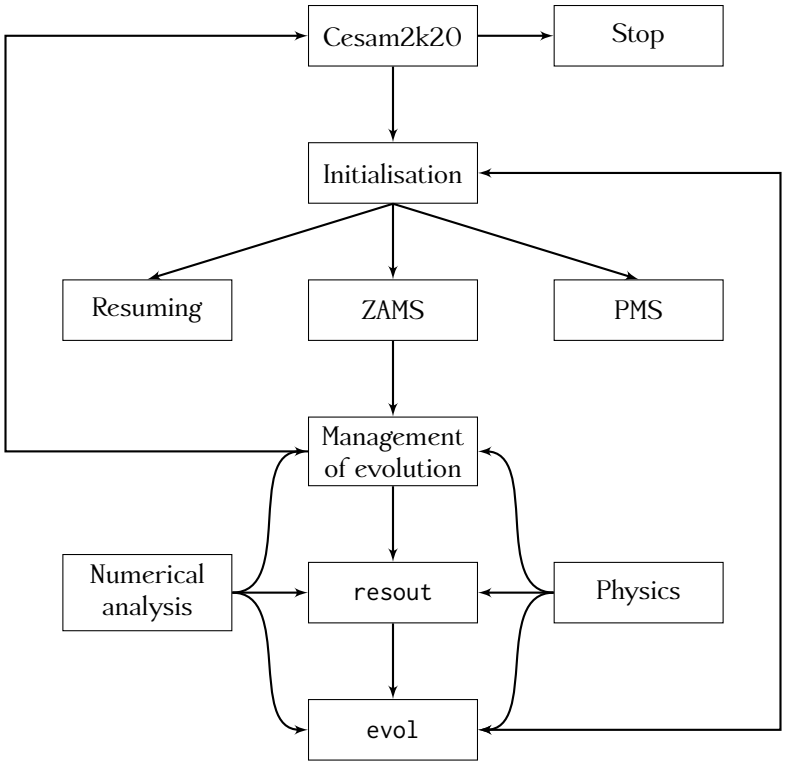


Figure 1.1: Diagram of the general flowchart of Cesam2k20. The program Cesam2k20, compiled once and for all, calls the routine `cesam` which is in fact the main program. In `cesam`, after initializations, we separate the treatment of the 3 possibilities: *i)* continuation of an evolution, *ii)* initialization on the homogeneous ZAMS, *iii)* initialization of a PMS. The calculation continues with operations for managing the evolution, listing, creating output files, drawing, etc. The next step is to use `resout`, where the quasi-static equilibrium equations are solved alternately with those of the evolution of angular momentum and chemical composition. The latter are solved separately in `evol`. Once convergence is achieved, there is a return to the evolution management algorithms of `evol` and from there, further evolution or return to `evol` for output. Numerical analysis and physics routines are used at different levels of calculation.

Chapter 2

Quick start guide

Contents

2.1 Installation

1. Clone the Cesam2k20 repository (ideally in your `home/username`):

```
git clone https://git.ias.u-psud.fr/joao.marques/cesam2k20_releases.git
mv cesam2k20_releases cesam2k20
cd cesam2k20
```

2. Then check you have all dependencies installed:

```
./check_deps.sh
```

   - The Fortran compiler (Cesam2k20 is usually compiled with `gfortran` or `ifort` but it may work with others);
   - The Fortran libraries needed by Cesam2k20;
   - The shell and Python interpreters;
   - The Python packages.

   The program uses `ldconfig` to find which compiler is installed, libraries like `lapack`, `blas`, or interpreters like `bash`. If `ldconfig` is not installed, `check_deps.sh` will not find anything, while you may have the dependencies installed. In this case, tell your system where to find above mentioned programs by appending the appropriate paths to your `LD_LIBRARY_PATH`.
   If a package is optional, and you don't have it installed, a yellow `no` will be printed on screen, otherwise, it is red.
   What to do if you miss something?

   - **Python 3:** We *strongly* advise you to install a python package manager (like `anaconda`[1]). Such package manager is easily installable on any PC or server (even locally). Take this time, it will be much easier later.

[1] `https://www.anaconda.com/products/distribution`, see at the bottom of the web page.

- **Other programs:** In general, we would advise to install all the missing packages with your usual package manager (e.g. `apt` on Ubuntu, `homebrew` or `macports` on MacOS, etc.). However, in some cases, it is not possible, especially on servers. In this case, an easy solution is to install the MESA Software Development Kit[2]. You will find everything you need in this package. Run their installation procedure carefully.

3. Configure the Makefiles. Several configuration files are available, depending on the Fortran compiler you want to use. Let's say you want to use `gfortran`, then run

```
./configure.sh gfortran
```

Recently, we add to Cesam2k20 the possibility to write structures in HDF5 format. This is optional and for that you need to have the `hdf5` library installed. If it is the case, then you can run

```
./configure.sh gfortran_h5
```

A configuration file also exists for `ifort`, but may need updating.

4. Now, you must say to your system where are the executables. Add following lines to your `.bashrc`, located in your `home`:

```
export CESDIR=${HOME}/cesam2k20

if [ -z ${LD_LIBRARY_PATH} ]
then
    export LD_LIBRARY_PATH="${CESDIR}/lib"
else
    export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:${CESDIR}/lib"
fi

export PATH="$PATH:${CESDIR}/bin/"
```

5. Finally, run

```
make install
```

2.1.1 Known issues

- As Cesam2k20 uses submodules, it needs a version of gfortran > 6.0 of ifort > 16.0. Check the compiler version with:

```
sudo gfortran --version
gfortran --version
```

(replace `gfortran` with your actual compiler). Careful: when you call `sudo`, your interpreter will not necessarily call the same `gfortran` than when you call `gfortran` directly. Are you calling the compiler you think you're calling ?

[2]MESA SDK: `http://user.astro.wisc.edu/~townsend/static.php?ref=mesasdk`

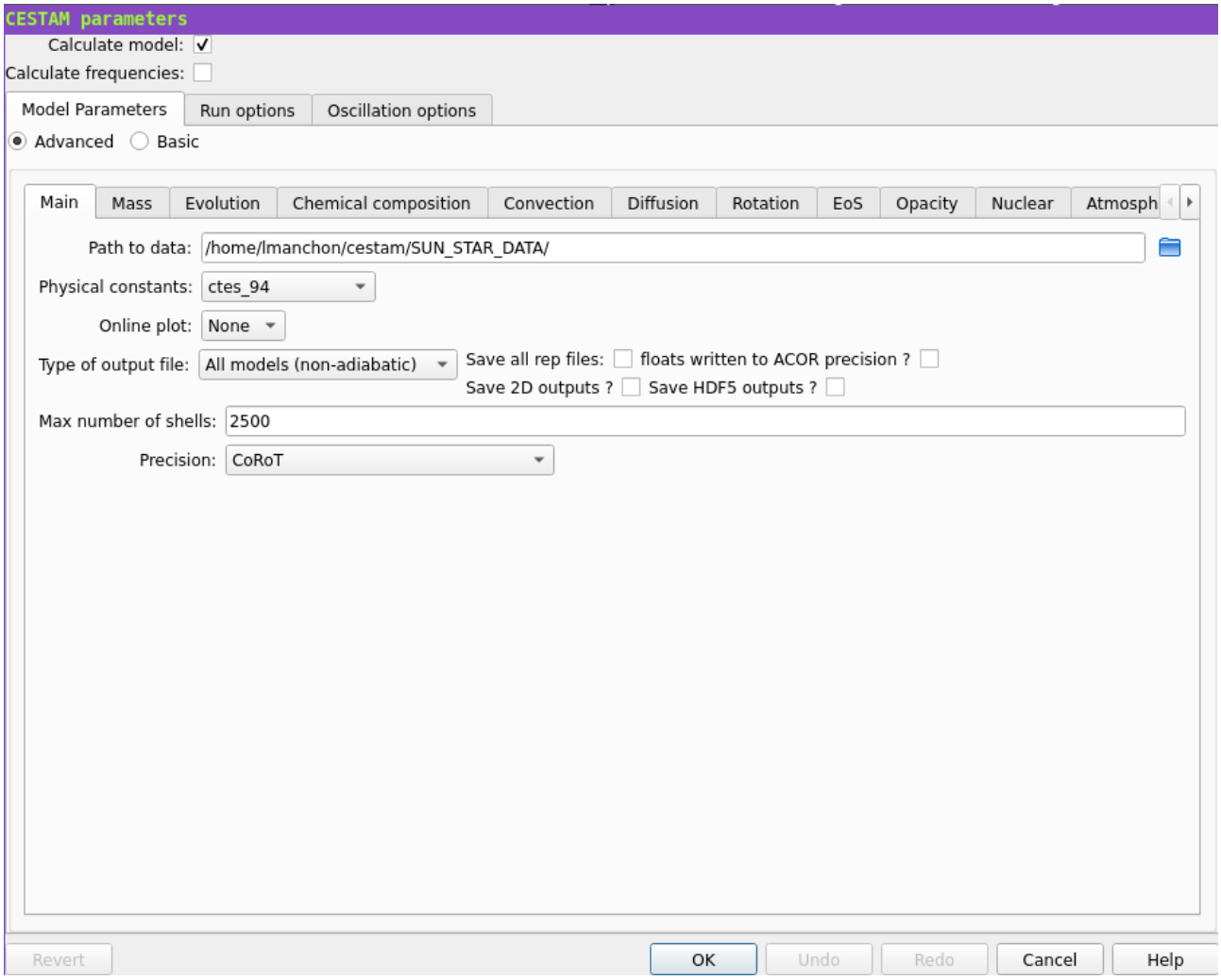


Figure 2.1: Home panel of the Graphic User Interface

- If after installation of MESA SDK, `gfortran` cannot find a needed library (e.g. `liblapack`), please prepend the right path to the `$LIBRARY_PATH` variable (not `$LD_LIBRARY_PATH`). For instance:

```
export LIBRARY_PATH="/home/username/mesasdk/math-slots/default/lib/:$LIBRARY_PATH"
```

2.2 First test: a Sun

For a quick test, start by creating a new folder (outside of the Cesam2k20 repository), e.g. called `sun`. Go in this repository, then run

```
run_cesam2k20.py model
```

`run_cesam2k20.py` is the name of the python script that will launch the Graphic User Interface, used to easily run Cesam2k20 models. `model` is the name of your model. You can give it any name you want, limited to 32 characters.

Once executed, the script opens a window similar to what appears in Fig. 2.1. You can choose between three tabs:

- 'Model parameters': Allows you to choose all the options in an input file called a `.don` file[3] This file will be automatically created once you click OK.
- 'Run options': Allows you to choose the starting point of the model. You can start from the PMS, the ZAMS, or resume an evolution[4]
- 'Oscillations options': Allows you to set options for the computation of oscillations. Oscillations are computed only if the box 'Calculate frequencies' in the top left corner is ticked.

Let us focused on the first tab 'Model parameters'. By default, initial conditions are preset in order to obtain a good model of the Sun at solar age. You can click 'Ok' and wait for the model to compute. It should take 3-4 min.

In the meantime, you can check the file `model.don` that have been created:

```
&NL_CESAM
NOM_CHEMIN = '/home/lmanchon/cesam2k20/SUN_STAR_DATA/', <= different in your case
NOM_CTES = 'ctes_94',
NOM_DES = 'no_des',
NOM_OUTPUT = 'osc_nadia',
NOM_OSC = 'none',
ACOR = F,
N_MAX = 2500,
PRECISION = 'co'
/
&NL_MASS
MTOT = 1.0,
NOM_PERTM = 'pertm_ext',
NOM_PERTM_SOLAR = 'pertm_ext',
NOM_PERTM_RGB = 'pertm_ext',
NOM_PERTM_AGB = 'pertm_ext',
NOM_PERTM_HOT = 'pertm_ext',
MDOT = 0.0
/
&NL_EVOL
AGEMAX = 4570.0,
ARRET = 'else',
DTLIST = 1000000000000.0,
LOG_TEFF = 10.0,
NB_MAX_MODELES = 10000,
HE_CORE = -1.0,
T_STOP = 50000000000.0,
X_STOP = -1.0,
XCX0 = -1.0,
RHOC = -1.0,
L_STOP = 10000000000.0,
R_STOP = 10000000000.0
```

[3] `don` stands for 'données' in French, 'data' in English.

[4] To resume an evolution, Cesam2k20 initiates the evolution from a previously generated structure model, stored in a `.rep` file. `rep` means 'reprise' in French, 'resumption' in English.

```
/
&NL_CHIM
GRILLE_FIXE = F,
NOM_ABON = 'solaire_ags09',
MODIF_CHIM = F,
GARDE_XISH = F,
X0 = 0.7311570588227411,
Y0 = 0.255442941177259,
ZSX0 = 0.018327115683702433
/
&NL_CONV
NOM_CONV = 'conv_jmj',
NOM_ALPHA = 'alpha_f',
ALPHA = 1.642090699560724,
NOM_OVSHTS = 'ovshts_f',
OVSHTS = 0.0,
OVSHTI = 0.0,
JPZ = F,
CPTURB = 0.0,
LEDOUX = F
/
&NL_DIFF
DIFFUSION = F,
NOM_DIFFM = 'diffm_mp',
NOM_DIFFT = 'difft_cte',
NOM_DIFFT_FING = 'no_fing',
NOM_DIFFT_SMC = 'no_smc',
D_TURB = 10.0,
RE_NU = 0.0,
NOM_FRAD = 'no_frad'
/
&NL_ROT
W_ROT = 0.0,
UNIT = 'kms/s',
NOM_DIFFW = 'diffw_0',
NOM_DIFFMG_TS = 'diffmg_ts_0',
NOM_GSF = 'no_gsf',
NOM_THW = 'rot_0',
ROT_ZC = 'solide',
NOM_PERTW = 'pertw_0',
P_PERTW = 6.5e+47,
LIM_JPZ = T,
MIXED_MODES = F,
INIT_ROT = F,
NOM_DES_ROT = 'no_des',
TAU_DISK = 0.0,
P_DISK = 0.0,
W_SAT = 8.0
/
```

```
 &NL_ETAT
 NOM_ETAT = 'etat_opal5Z',
 F_EOS = 'eos5_sunags09_nodiff.bin', '', '', '', '', '', '', ''
 /
 &NL_OPA
 NOM_OPA = 'opa_yveline',
 NOM_OPA_COND = 'cond_yveline21',
 F_OPA = 'opa_yveline_AGSS09_met.bin', 'TOPS_MW', 'TOPS_LMC', 'TOPS_SMC', '', '', '', ''
 /
 &NL_NUC
 NOM_NUC = 'nrj31_YL',
 NOM_NUC_CPL = 'NACRE_LUNA',
 NOM_NEUTRINOS = 'neutrinos_h94w66',
 MITLER = F
 /
 &NL_ATM
 NOM_ATM = 'lim_atm',
 NOM_TDETAU = 'hopf',
 TAU_MAX = 20.0,
 LIM_RO = T
 /
```

The .don file is a Namelist. All entries are described in Chapter 3.

When the computation is finished, you can quickly explore the evolutionary track or the structure models. For that, use the command:

```
plot_hr.py model
```

to plot an HR diagram, or

```
plot_osc.py model
```

to obtain an interactive plot that allow you to plot any structural quantity against any other one.

More details are given Part. III

Chapter 3

Advanced setup

Va despacio, estoy de prisa.

Proverbe Mexicain.

(Je vais doucement car je suis pressé)

Contents

3.1 Advices for the modeller

A digital code, however well done, cannot be "turnkey". To start an exploitation, one must always be guided by the trivial remark:

A programme that does not work, or works badly, for simple cases will not work, or work even worse, for complicated cases.

An internal structure code is, unfortunately, no exception to this rule. To this end, although Cesam2k20 has provided for various types of standard 'precision', it is almost always necessary, with the help of 'adjustments', to optimise the numerical parameters according to the conditions encountered during the calculation. It is a waste of time to approach a problem by introducing *ad initio* all the complexity one wishes to achieve. In order to gain experience on how the evolution unfolds, it will be necessary to take up simple cases very quickly. It is advisable to "de-coarse" the problem using robust and efficient numerical and physical options, in order to quickly appreciate "what is going on", and then to gradually introduce the desired complexity.

The descriptions of the routines given above are only brief, see the comments on the routines for more information.

3.2 The input file: *.don

The physical options to be used and the data are transmitted to the main program by means of NAMELISTs, grouped in an ASCII file whose name has the extension .don, for example: model.don. An example of this file is located in the subdirectory exploit.

- NL_CESAM: gathers general information for the computation.
- NL_MASS: gathers physical parameters concerning the mass variable.
- NL_EVOL: gathers physical parameters concerning the time variable.
- NL_CHIM: gathers physical parameters concerning chemical composition.
- NL_CONV: gathers physical parameters concerning convection.
- NL_DIFF: gathers physical parameters concerning diffusion.
- NL_ROT: gathers physical parameters concerning rtation.NL_ROT
- NL_ETAT: gathers file names accessed *externally* necessary for EoS computation.
- NL_OPA: gathers file names accessed *externally* necessary for opacity computation.
- NL_NUC: gathers physical parameters concerning nuclear reaction rates.
- NL_ATM: gathers physical parameters concerning the restitution of atmosphere.
- NL_NUM: gathers numerical parameters for fine tuning of numerical methods.
- NL_RLG: gathers parameters for fine tuning of physical models.

3.3 Example of .don file

This example is deliberately different from the one used in Sect. 2.2.

```
&NL_CESAM
NOM_CHEMIN = '/home/lmanchon/cesam2k20/SUN_STAR_DATA/',
NOM_CTES = 'ctes_94',
NOM_DES = 'no_des',
NOM_OUTPUT = 'osc_nadia',
NOM_OSC = 'none',
ACOR = F,
```

```
N_MAX = 2500,
PRECISION = 'co'
/
&NL_MASS
MTOT = 1.0,
NOM_PERTM = 'pertm_ext',
NOM_PERTM_SOLAR = 'pertm_ext',
NOM_PERTM_RGB = 'pertm_ext',
NOM_PERTM_AGB = 'pertm_ext',
NOM_PERTM_HOT = 'pertm_ext',
MDOT = 0.0
/
&NL_EVOL
AGEMAX = 4570.0,
ARRET = 'else',
DTLIST = 1000000000000.0,
LOG_TEFF = 10.0,
NB_MAX_MODELES = 10000,
HE_CORE = -1.0,
T_STOP = 50000000000.0,
X_STOP = -1.0,
XCX0 = -1.0,
RHOC = -1.0,
L_STOP = 10000000000.0,
R_STOP = 10000000000.0
/
&NL_CHIM
GRILLE_FIXE = F,
NOM_ABON = 'solaire_ags09',
MODIF_CHIM = F,
GARDE_XISH = F,
X0 = 0.7311570588227411,
Y0 = 0.255442941177259,
ZSX0 = 0.018327115683702433
/
&NL_CONV
NOM_CONV = 'conv_jmj',
NOM_ALPHA = 'alpha_f',
ALPHA = 1.642090699560724,
NOM_OVSHTS = 'ovshts_f',
OVSHTS = 0.0,
OVSHTI = 0.0,
JPZ = F,
CPTURB = 0.0,
LEDOUX = F
/
&NL_DIFF
DIFFUSION = F,
NOM_DIFFM = 'diffm_mp',
```

```
NOM_DIFFT = 'difft_cte',
NOM_DIFFT_FING = 'no_fing',
NOM_DIFFT_SMC = 'no_smc',
D_TURB = 10.0,
RE_NU = 0.0,
NOM_FRAD = 'no_frad'
/
&NL_ROT
W_ROT = 0.0,
UNIT = 'kms/s',
NOM_DIFFW = 'diffw_0',
NOM_DIFFMG_TS = 'diffmg_ts_0',
NOM_GSF = 'no_gsf',
NOM_THW = 'rot_0',
ROT_ZC = 'solide',
NOM_PERTW = 'pertw_0',
P_PERTW = 6.5e+47,
LIM_JPZ = T,
MIXED_MODES = F,
INIT_ROT = F,
NOM_DES_ROT = 'no_des',
TAU_DISK = 0.0,
P_DISK = 0.0,
W_SAT = 8.0
/
&NL_ETAT
NOM_ETAT = 'etat_opal5Z',
F_EOS = 'eos5_sunags09_nodiff.bin', '', '', '', '', '', '', ''
/
&NL_OPA
NOM_OPA = 'opa_yveline',
NOM_OPA_COND = 'cond_yveline21',
F_OPA = 'opa_yveline_AGSS09_met.bin', 'TOPS_MW', 'TOPS_LMC', 'TOPS_SMC', '', '', '', ''
/
&NL_NUC
NOM_NUC = 'nrj31_YL',
NOM_NUC_CPL = 'NACRE_LUNA',
NOM_NEUTRINOS = 'neutrinos_h94w66',
MITLER = F
/
&NL_ATM
NOM_ATM = 'lim_atm',
NOM_TDETAU = 'hopf',
TAU_MAX = 20.0,
LIM_RO = T
/
```

The names of the routines are *case sensitive*, for example, it is necessary to code `"NACRE"` and not `"nacre"`. All parameters in `NL_NUM` and `NL_RLG` are optional because they all have a default value set

inside Cesam2k20.

3.4 NL_CESAM namelist members

In the following, we write name list members in capital letters for the first time, then in lower case, which refer to the variable in the code.

- NOM_CHEMIN (character): path and name of the directory where the physical data, e.g. the opacity tables, are stored.
- NOM_CTES (character): indicates the name of the initialization routine for the main physics constants to be used:
  - 'ctes_85': Constants from GONG Christensen-Dalsgaard (1988), NACRE and Handbook of Chemistry and Physics (Lide, 1994). Solar values are taken from Guenter et al (???). Solar [Fe/H]is from Grevesse & Noels (1993).
  - 'ctes_94': Same as above but GONG values are not used
  - 'ctes_94m': Identical to ctes_94 with values of the masses of the nuclei set equal to the nearest integer values.
  - 'ctes_94_asplund': Identical to ctes_94 with Solar metallicity from (Asplund et al., 2009).
  - 'ctes_gs98': Physical constants are taken from the PLATO WP120 technical notes on physical constants, gathered from IAU 2015 and CODATA 2018 resolutions. Solar metallicity from (Grevesse & Sauval, 1998). The values of other constants can be reverted to the same as in ctes_94 (except $[\mathrm{Fe/H}]_\odot$) by setting extra%source_cts = 'Lide1994'. otherwise, default source is extra%source_cts = 'IAU2015_CODATA2018'.
  - 'ctes_aag21': Physical constants are taken from the PLATO WP120 technical notes on physical constants, gathered from IAU 2015 and CODATA 2018 resolutions. Solar metallicity from (Asplund et al., 2021). The values of other constants can be reverted to the same as in ctes_94 (except $[\mathrm{Fe/H}]_\odot$) by setting extra%source_cts = 'Lide1994'. otherwise, default source is extra%source_cts = 'IAU2015_CODATA2018'.
- NOM_DES (character): name of the drawing routine to be used (**Depreciated**: here for historical reasons, routines not available anymore; you should use the pythons package now):
- NOM_OUTPUT (character): type of ASCII file to be generated at the end of the calculation:
  - 'osc_adia: Option Generation of the ASCII file with the name *-ad.osc for the calculation of the adiabatic oscillations and the realization of certain drawings.
  - 'osc_invers: Generation of ASCII file of name *-inv.osc for inversions.
  - 'osc_nadia: Generation of the ASCII file with the name *-nad.osc for the calculation of non-adiabatic oscillations.
  - 'osc_plato: Generation of the ASCII file with the name *-plato.osc, specially designed for PLATO grids, with only quantities necessary for adiabatic oscillation computation, and X, Y and Z profiles.
  - 'no_output': No ASCII file generation.
  - **Simulfix 'all'**: You can replace the first three characters 'osc' with 'all' (e.g. 'osc_adia' becomes all_adia). In this case, not only the final structure is stored into a file, but the structures at all time steps. In this case, a 5-digit (0-prefixed) if added before the name of each osc file (even the last one), e.g. nnnnn*-ad.osc.

  - **Suffixes `'2d'` `'h5'`, `'2dh5'`**: You can add these suffixes after any of previous options, e.g. `'osc_adia2d'`, `'osc_adiah5'` or `'osc_adia2dh5'`. With suffixes `'2d'` or `'2dh5'`, Cesam2k20 generates an extra `.osc2d` file. With suffixes `'h5'` or `'2dh5'`, Cesam2k20 generates a `*.osch5` HDF5 file *in replacement* of the `*.osc` file (e.g. `*-ad.osch5` instead of `*-ad.osc`). In this case, if the prefix `'all'` is used instead of `'osc'`, the structures of all time steps will be stored in the same `osch5` file. Therefore, the integer `nnnnn` is not prepended anymore.

- `N_MAX (integer)`: the maximum number of layers [1]. The value of `n_max` specified in the `.don` file can be overwritten by the one specified by predefined choice of precision parameters (see below), if this number is higher. At the same time, a minimum number of layers is set at 300. Provided that the formation of an ASCII file is required, a negative value of `n_max` allows the last model of the evolution to be calculated with a number of layers equal to |`n_max`|.

- `PRECISION (character)`: level of precision required. The values of the numerical parameters used according to the different options defined in the `cesam` routine are found in Table 3.1. The available values are:

  - `'pr'`: (Realistic precision) For an evolution without a particular search for precision, with Lagrangian coordinates.
  - `'er'`: (Realistic precision) For an evolution without a particular search for precision, with Eulerian coordinates.
  - `'sp'`: (Super precision) If extreme precision is required. Lagrangian coordinates are used.
  - `'sr'`: (Super precision) If extreme precision is required. Eulerian coordinates are used.
  - `'av'`: (Advanced stages) For an evolution towards advanced stages without a particular search for precision.
  - `'co'`: (CoRoT precision) Precision tuning devised at the time of CoRoT mission.
  - `'pl'`: (PLATO precision) Precision tuning devised for thePLATO mission. The idea is to obtain super precision and an equival:ent number of models for any stellar mass at a given evolutionary status.
  - `'mj'`: (Marie-Jo's precision) Specially designed for Marie-Jo Goupil.
  - `'lm'`: (Low-mass precision) Precision for very evolved low-mass stars.
  - `'sa'`: (Solar accuracy) Similar to `'sp'` but the last models of an evolution are calculated with the maximum number of layers `N_MAX`.
  - `'hp'`: (Hyper precision) Even better than super precision.
  - `'np'`: (Normal precision) Useful to get the general idea of an evolution. Lagrangian coordinates are used.
  - `'nr'`: (Normal precision) Useful to get the general idea of an evolution. Eulerian coordinates are used.
  - `'mx'`: (Maximum of layers) identical to `'sp'` the calculation being done with a *fixed* number of layers, equal to `N_MAX`.

Impact on numerical parameters are summarized in Sect. 3.16.

[1] The nature of the numerical and logical parameters is identifiable by the values given for guidance.

3.5 NL_MASS namelist members

- MTOT (real): initial total mass, in $M_\odot$.
- NOM_PERTM (character) : name of the mass loss routine. Kept for retro compatibility. If this option is used. It will be applied at every phases or for any temperature. **Depreciated**: use specific mass loss prescription instead (see following points.) Available prescriptions:
  - 'pertm_0' : No mass loss. pertm_0.f90
  - 'pertm_ext' : Linear external mass loss/gain as a function of time. pertm_ext.f90
  - 'pertm_msol' : Linear external mass loss/gain as a function of time, with the mass of the star remaining at least equal to 1 $M_\odot$. pertm_msol.f90
  - 'pertm_tot' : External mass loss/gain, linear with time and taking into account the mass loss due to thermonuclear reactions ($E = mc^2$). pertm_tot.f90
  - 'pertm_reimers' : Empirical mass loss of Reimers (1975). pertm_reimers.f90
  - 'pertm_waldron' : Empirical mass loss of Waldron (1990). pertm_waldron.f90
  - 'pertm_b_1994' : Empirical mass loss of Blöcker (1994). pertm_B_1994.f90
  - 'pertm_cs_2011' : Empirical mass loss of Cranmer and Saar (2011). pertm_CS_2011.f90
  - 'pertm_vanloon' : Empirical mass loss of van Loon et al. (2005). pertm_vanLoon.f90
  - 'pertm_jnh_1987' : Empirical mass loss of de Jager et al. (1987). pertm_JNH_1987.f90
  - 'pertm_nj_1990' : Empirical mass loss of Nieuwenhuijzen and de Jager (1990). pertm_NJ_1990.f90
  - 'pertm_vink' : Empirical mass loss of Vink et al. (2001). pertm_vink.f90
- NOM_PERTM_SOLAR (character) : prescription used for models with $T_{\rm eff} < 10^4$ K, in the main sequence. Prescriptions allowed: 'pertm_0', 'pertm_msol', 'pertm_tot'.
- NOM_PERTM_RGB (character) : prescription used for models with $T_{\rm eff} < 10^4$ K, in post evolution phases (sub- and red giants). Prescriptions allowed: 'pertm_0', 'pertm_ext', 'pertm_reimers', 'pertm_vanLoon'.
- NOM_PERTM_AGB (character) : prescription used for models with $T_{\rm eff} < 10^4$ K, where burning of helium, carbon or oxygen started in the core. Prescriptions allowed: 'pertm_0', 'pertm_ext', 'pertm_reimers', 'pertm_vanLoon'.
- NOM_PERTM_HOT (character) : prescription used for models with $T_{\rm eff} > 10^4$ K. Prescriptions allowed: 'pertm_0', 'pertm_ext', 'pertm_vink'.
- MDOT (real): mass gain/loss rate, in $M_\odot \cdot \text{year}^{-1}$, this rate is positive for a mass gain, negative for a mass loss. In the standard case, the chemical composition of the mass gain/loss is that of the outermost layer of the model. See Sect. 3.15.4 how to customise this chemical composition, if necessary.

3.6 NL_EVOL namelist members

- AGEMAX (real): maximum age to be reached in millions of years.
- ARRET (character): stops on the ZAMS ('zams'), at the end of the ZAMS ('post'[2]), at the beginning of the combustion of helium ('cohe') or carbon ('coca') or oxygen ('coox') or other ('else') (see description below 3.6.1).

[2] 'tams' is accepted.

- DTLIST (real): minimum time interval, in million years, between two detailed lists of the model (file *.lis). **Depreciated**.
- LOG_TEFF (real): stops if this value of $\log_{10} T_{\rm eff}$ is crossed, in the increasing direction if log_teff is positive, in the decreasing direction if log_teff is negative.
- NB_MAX_MODELES (integer): stops after calculating the number of models indicated, possibly 0; with nb_max_modeles < 0, all the binary files will be written in the environment, their names including the model number and the identification extension, *.dat, *.rep, etc. Example: sun0145_B.dat. The number nnnn is assigned to ZAMS or PMS initialisation models; in case of takeover, the numbers start from the one of the taken over model.
- HE_CORE (real): stops when the mass of the helium core reaches the value he_core (in unit of $M_\odot$).
- R_STOP (real): stops when the value of the bolometric radius crosses r_stop (in unit of $R_\odot$). If r_stop < 0: stops when luminosity decreases below |r_stop|, else luminosity increases above r_stop.
- L_STOP (real): stops when the value of the bolometric radius crosses l_stop (in unit of $L_\odot$). If r_stop < 0: stops when luminosity decreases below |l_stop|, else luminosity increases above l_stop.
- T_STOP (real): stops if the temperature in the centre exceeds this value.
- X_STOP (real): stops if this value of X, in the centre, is crossed.
- XCX0 (real): stops if this percentage of initial hydrogen abundance is reached in the centre.
- RHOC (real): stops if this central density is reached.

3.6.1 Characterisation of evolutionary stage

- 'zams': A model describing the main pre-sequence is identified with a ZAMS model as soon as the nuclear energy flow exceeds that of gravitic origin.
- 'post': A model describing the main sequence is identified with a post-main sequence model as soon as the central hydrogen abundance falls below 0.01.
- 'cohe': A model describing the main post-sequence is identified with a helium burning model as soon as the central temperature exceeds $1 \cdot 10^8$K.
- 'coca': A model describing the helium burning sequence is identified with a model with carbon burning as soon as the central temperature exceeds $6 \cdot 10^8$K.
- 'coox': A model describing the carbon combustion sequence is identified with a model with oxygen combustion as soon as the central temperature exceeds $1 \cdot 10^9$K.

3.7 NL_CHIM namelist members

- GRILLE_FIXE (logical): a fixed grid will be used for chemical composition and momentum diffusion.
- NOM_ABON (character): name of initial mixture (in **green**, mixtures for which a dedicated opacity table is available with Cesam2k20[3]):

[3] see EoS OPAL: http://www-phys.llnl.gov/V\char'_Div/OPAL/opal.html.

  - 'enhan_al': Allard's α-enhanced abundances.
  - 'enhan_cha': Chaboyer's α-enhanced abundances.
  - 'enhan_w': Weiss's α-enhanced abundances.
  - 'meteorites_ag': meteoritic abundances of Anders & Grevesse (1989).
  - 'meteorites_gs': meteoritic abundances of Grevesse & Sauval (1998).
  - 'meteorites_ags05': meteoritic abundances of Asplund et al. (2005).
  - 'mixture': custom form of the abundance compilation described in Sect. 3.15.1.
  - 'solaire_gn': solar mixture of Grevesse & Noels (1993).
  - 'solaire_gs': solar mixture of Grevesse & Sauval (1998).
  - 'solaire_ags05': solar mixture of Asplund et al. (2005).
  - 'solaire_ags05i': solar mixture of Asplund et al. (2005) with initial lithium.
  - 'solaire_ags09': solar mixture of Asplund et al. (2009) following Serenelli et al. (2009) recommendation.
  - 'solaire_ags09_0': original solar mixture of Asplund et al. (2009). So-called photospheric mixture (exceptions: meteoritic Li, Be, B, F)
  - 'solaire_ags09_SBBN': solar mixture of Asplund et al. (2009) with SBBN lithium (Coc & Vangioni, 2017).
  - 'solaire_ags09_alpha': solar mixture of Asplund et al. (2009) with α-enrichment following Deal et al. (2020).
  - 'solaire_aag21': solar mixture of Asplund et al. (2021).

  These abundances are initialized in the routine abon_ini.

- MODIF_CHIM (logical): if, in the environment, files of the types modif_mix, rap_iso, planet or vent exist which allow, respectively, to modify the initial chemical composition (see Sect. 3.15.3), the isotope ratios (see Sect. 3.15.2), the chemical composition of the planetoids (see Sect. 3.15.5) or that of the wind (see Sect. 3.15.4), Cesam2k20 will invite you to delete these files before performing the calculations. This provision, *for safety purpose*, is intended to avoid taking into account files which, by mistake, would not have been subtracted from the environment. It is advisable to code modif_chim=.true. to take into account the provisions offered by the presence of these files.
- GARDE_XISH (logical): This parameter is discussed in the next paragraph.
- X0 (real): initial abundance, per unit mass, of H.
- Y0 (real): initial abundance, per unit mass, of He.
- ZSX0 (real): Initial value of Z/X.
  - The abundances of the initial model are determined from the quantities x0, y0 and zsx0; because of the relation $X + Y + Z = 1$, only 2 of these 3 quantities are to be taken into consideration. The initial value of Y used in the calculation is *always* y0; it is split into the various isotopes retained to describe the evolution of helium.
  - if zsx0 ≤ 0, the initial hydrogen abundance is initialized to x0, $Z = 1 - X - Y$ is deduced from x0 and y0.
  - otherwise, i.e. zsx0 > 0, the initial hydrogen abundance is deduced from y0 and zsx0, the value indicated for x0 is ignored.

The initial value of X used in the calculation is split into the various isotopes used to describe the evolution of hydrogen.

With diffusion and/or after a first dredge-up, the chemical composition of the outer layers, i.e. the observed one, differs from the initial chemical composition. The initial value of the chemical composition has to be adjusted in order to obtain the observed value after an evolution. Observations on the metal/H, or [Fe/H], ratios, initiated by F. Thévenin, offer various tools for making this adjustment:

- Keep the ratio Z/X of the initial mixture, instead of `ZSX0`, this is discussed in Sect. 3.7.1.
- Use an original mixture (see Sect. 3.15.1).
- Change the isotope ratios (see Sect. 3.15.2).
- Change the abundance ratios of the initial mixture (see Sect. 3.15.3).

3.7.1 Alternative: metal/H or metal/Z conservation

For an initial mixture, possibly modified or not, see Sect. 3.15.3, the alternative `garde_xish=.true.` (respectively `garde_xish=.false.`) allows the initial value of Z to be fixed so as to maintain the ratio of metal/H (respectively Metal/Z). The initial value of Y is *always* the value read from the data file, i.e. `y0`.

Thus, by coding[4] `garde_xish=.true.`, the value of Z/X used will be deduced from the metal/H abundance ratios of the mixture and not the one, `zsx0`, read in the data file[5]. The M/H abundance ratios will be those of the mixture, the metal/Z ratios will be different from those of the mixture.

Problem: *The use of* `garde_xish=.true.` *is delicate. We advise to check in the standard output that the initial abundances values used by Cesam2k20 correspond to what you wanted.*

Problem: *You must keep in mind that modifications to the mixture are not taken into account in the opacity and EoS tables.*

3.8 NL_CONV namelist members

- `NOM_CONV (character)`: Name of the routine computing the temperature gradient in the convective zones.
  - `'conv_a0'`: MLT convection with mixing length $\ell_{\rm MLT}$ going to 0 at limits RZ/CZ, following prescription of Eggleton (1972).
  - `'conv_cgm_reza'`: Convection following Canuto et al. (1996), avec $\ell = \ell_{\rm CGM} H_p$, accounting for the prescription of Bernkopf.
  - `'conv_cm'`: Convection following Canuto & Mazzitelli (1991), avec $\ell = \ell_{\rm CM} H_p$.
  - `'conv_cm_reza'`: Convection following Canuto & Mazzitelli (1991), avec $\ell = \ell_{\rm CM} H_p$, accounting for the value of δ,
  - `'conv_jmj'`: MLT convection with mixing length $\ell = \alpha_{\rm MLT} H_p$ (Böhm-Vitense, 1958).
- `NOM_ALPHA (character)` Name of the routine to make the α parameter vary along evolution.
  - `'alpha_f'`: α stays fixed along evolution.
  - `'entropy_ludwig99'`: α is adjusted so that entropy of the adiabat in the 1D model matches the prescription of Ludwig et al. (1999).

[4] Better names would be, for example: `gard_xish_mix`, `garde_xish_mix_ini`, `garde_zsx` or `garde_zsx_mix_ini`,etc.
[5] Except in the particular case where the values of `y0` and `zsx0` of the data file correspond *exactly* to those of the mixture

  - 'entropy_magic13': α is adjusted so that entropy of the adiabat in the 1D model matches the prescription of Magic et al. (2013).
  - 'entropy_tanner16': α is adjusted so that entropy of the adiabat in the 1D model matches the prescription of Tanner et al. (2016).
  - 'entropy_custom': α is adjusted so that entropy of the adiabat in the 1D model matches a custom prescription (hard coded). It is advised not to use this one as it is intended just to perform tests.

- ALPHA (real): Convection parameter, either $\alpha_{\rm MLT}$, $\ell_{\rm CGM}$ or $\ell_{\rm CM}$.
- NOM_OVSHTS (character): Formalism used to model the overshoot.
  - 'ovshts_f': The overshoot parameters ovshts stays fixed along evoltuion. Core convective zone extended by ovshts $\times H_p$
  - 'ovshts_claret18':
  - 'ovshts_f_diff':
  - 'ovshts_claret18_diff':
- OVSHTS (real): Upper overshoot coefficient. Contrary to previous version of CESAM, the sign of ovshts do not influence the gradient used in the overshoot region. The gradient used is set through extra%ovs_type and ovshts is set to its absolute value.
- OVSHTI (real): Lower overshoot coefficient. Contrary to previous version of CESAM, the sign of ovshti do not influence the gradient used in the overshoot region. The gradient used is set through extra%ovi_type and ovshti is set to its absolute value.
- JPZ (logical): Treat overshoot regions according to (Zahn, 1991). With the introduction of new overshoot gradient prescription in Cesam2k20, the use of Zahn (1991)'s treatment of overshoot regions is slightly different than the one presented in the original paper. If you want to have a true (Zahn, 1991)'s treatment, use `jpz = .true.` and `extra%ovs_type = 'adia'`.
- CPTURB (real): Turbulent pressure coefficient.
- LEDOUX (logical): Use of Ledoux criteria for convective instability. If the semi-convection is taken into account (see 3.9), ledoux must be set to false, because the Ledoux formalism is part of the semi-convection description.

Problem: *U*sing a non-zero turbulent pressure parameter is tricky.

3.9 NL_DIFF namelist members

- DIFFUSION (logical): microscopic diffusion of chemical elements and possibly angular momentum diffusion will be taken into account,
- NOM_DIFFM (character): name of the routine calculating the microscopic diffusion coefficients:
  - 'diffm_br': calculation of microscopic diffusion coefficients according to Burgers (1969)' formalism, possibly taking radiative accelerations into account,
  - 'diffm_mp': calculation of the microscopic diffusion coefficients using the simplified Michaud & Proffitt (1993) formalism,
  - 'diffm_0': zero microscopic diffusion coefficients.
- NOM_DIFFT (character): name of the routine calculating the turbulent diffusion coefficients:

- 'difft_c0': includes no additional turbulent diffusion coefficient.
- 'difft_cte': includes a constant diffusion coefficient defined by D_TURB in $cm^2.s^{-1}$.
- 'difft_expo': include a double exponential decrease of the turbulent diffusion coefficient

$$D_{\rm turb} = \omega_1 \exp\left(\frac{(r - r_{\rm cz})}{R_\star \sigma_1} \ln 2\right) + \omega_2 \exp\left(\frac{(r - r_{\rm cz})}{R_\star \sigma_2} \ln 2\right), \tag{3.1}$$

  where, ω_1 and ω_2 are defined by D_TURB and extra%D_expo_2, respectively. σ_1 and σ_2 are defined by extra%L_expo and extra%L_expo_2, respectively. $r_{\rm cz}$ is the radius of the base of the convective envelope.
- 'difft_montreal': includes a profile for the turbulent diffusion coefficient following Richer et al. 2000: $D_{\rm turb} = \omega D({\rm He})_0 \, (\rho_0/\rho)^{\rm n}$ where ω and n are constants defined by D_TURB and extra%expo_mix, respectively. ρ and $D({\rm He})$ are the local density and diffusion coefficient of helium. The indices 0 indicate that the value is taken at a reference depth. The reference depth can be defined either in mass of the envelope (according to the total mass of the star) or in temperature by the parameter extra%M_mix and extra%T_mix, respectively.
- 'difft_montreal_cz': includes profiles for the turbulent diffusion coefficient following Richer et al. 2000 imposing the reference depth to the base of the convective envelope. ω and n are constants defined by D_TURB and extra%expo_mix, respectively.
- 'difft_grille': includes the expression of Richer et al. 2000 but with two different values depending on the effective temperature at the ZAMS for the model. Between the two points in the grid, a linear variation in depth of the mixing is applied. An example of the variation of the depth is given in Fig. **??**. ω and n are constants defined by D_TURB and extra%expo_mix, respectively. The reference depth can be defined only in temperature by the parameter extra%t0_Sun for the first point (generally the Sun is used as reference) and by extra%t0_2 for the second reference point. The selection points in effective temperature are defined by extra%teff_zams_Sun and extra%teff_zams_2.
- 'difft_gab': extends the convective mixing for $t < 10^6$ K.
- 'difft_sun': includes a specific turbulent diffusion coefficient profile for the Sun following Gabriel (1997)'s work.

  An extra constant turbulent diffusion coefficient can be added to difft_montreal, difft_montreal_cz and difft_grille with extra%cst_mix.

- NOM_DIFFT_FING (character) is the name of the thermohaline convection routine. There are two choises (they can be found in difft_fing.f90):
  - no_fing no thermohaline convection.
  - brown2013: include thermohaline mixing with the formalism of Brown et al. 2013.

  An extra output can be provided with the option num%th_out = .TRUE..
- NOM_DIFFT_SMC (character) is the name of the semi-convection routine. There are two choices (they can be found in difft_smc.f90):
  - 'no_smc': no semi-convection.
  - 'langer1983': include thermohaline mixing with the formalism of Langer et al. 1983.
- D_TURB (real): isotropic turbulent diffusion coefficient. It has a different meaning, depending on the turbulent diffusion model used (see above).

- RE_NU (real): radiative diffusivity coefficient, defined in Morel & Thevenin 2002. This coefficient is added if RE_NU>0. It has been shown by Alecian & Michaud 2005 that the physical meaning of this process was questionable. *We strongly advise not to include values different from* 0 *for this parameter.*

- NOM_FRAD (character) defines the radiative acceleration routines. There are three choices (they can be found in f_rad.f90):

  - 'alecian2004': includes Alecian & LeBlanc 2004 Single Valuate Parameters (SVP) method.
  - 'alecian2020': includes the updated Alecian & LeBlanc 2020 SVP method.
  - 'no_frad': includes no radiative accelerations.

  For radiative accelerations to be included the specific nuclear reactions nets in which grad appears in the name are required, namely NOM_NUC = 'ppcno9grad' and NOM_NUC = 'ppcno13gradNi'. The computation of the Rosseland mean opacity can be done with the OP monochromatic opacities to follow the variation of mixture induced by atomic diffusion with the option NOM_OPA = 'opa_mono_op'. *With this option, the computation time is drastically increased and a minimum of 1Go of RAM should be available for the tables to be loaded.* The control parameter extra%goop can be used to prevent the computation of radiative accelerations and the use of monochromatic opacity before a specific age given in Myr. Note that for the moment, radiative accelerations are automatically disabled after the TAMS for numerical stability and because the process is almost negligible at this evolutionary stage (see Moedas et al. 2022). Extra outputs can be provided for the diffusion velocities and radiative accelerations with the option num%vdiff_out = .TRUE. and num%grad_out = .TRUE., respectively.

3.10 NL_ROT namelist members

- W_ROT (real): initial angular velocity (It is advised not to use this option, but to initialize the rotation with the period and life time of the disk).

- UNIT (character): unit used for the initial angular velocity. Different choices are proposed to easily adapt to observations:

  - 'kms/s': velocity of the outer layer in kilometers per second, the radius of the initialization model being used to determine the initial angular velocity, it is necessary to adjust this radius by recalculating the model of age 0 several times,
  - 'rad/s': the initial angular velocity is in radians per second.

- NOM_DIFFW (character): name of the routine for calculating the diffusion coefficients of angular momentum:

  - 'diffw_0': zero diffusion coefficients of angular momentum.
  - 'diffw_cte': constant diffusion coefficients of angular momentum: ($D_{\rm eff} = 300, D_{\rm h} = 10^6, D_{\rm v} = 250$).
  - 'diffw_p03': diffusion coefficients of angular momentum according to Palacios et al. (2003).
  - 'diffw_mpz': diffusion coefficients of angular momentum according to Mathis et al. (2004).
  - 'diffw_mathis2018': diffusion coefficients of angular momentum according to Mathis et al. (2018). .

- NOM_DIFFMG_TS: name of the prescription used for AM transport by Tayler-Spruit (TS) instability:
  - 'diffmg_0': No TS instability;
  - 'diffmg_ts_spruit2002': Prescription of Spruit (2002).
  - 'diffmg_ts_maeder2004': Prescription of Maeder & Meynet (2004).
  - 'diffmg_ts_daniel2023': Prescription of Daniel et al. (2023).
- NOM_THW (character): Name of the theory used for the evolution of the angular velocity:
  - 'rot_0': Zero angular velocity, the model does not take rotation into account.
  - 'rot_cte': Constant angular velocity during evolution, i.e., rigid rotation.
  - 'cons_glob_mnt_cin': Evolution with rigid rotation, and global conservation of angular momentum. At the end of each time step, the angular velocity, constant throughout the model, is readjusted to conserve the total angular momentum.
  - 'cons_loc_mnt_cin': Evolution with local conservation of angular momentum with rigid rotation of the CZ.
  - 'diff_tz97': Evolution with angular momentum diffusion, following the theory of Talon et al. (1997).
  - 'diff_mz04': Evolution with angular momentum diffusion, following the theory of Mathis & Zahn (2004).
- ROT_ZC (character): Rotation in CZ:
  - 'solide': constant angular velocity;
  - 'local_j': constant angular momentum.
- NOM_PERTW (character): Name of the routine for calculating angular momentum loss.
  - 'pertw_0': No mass loss;
  - 'pertw_kaw': Prescription of Kawaler (1988);
  - 'pertw_matt15': Prescription of Matt et al. (2015);
  - 'pertw_rm': Prescription of Reiners & Mohanty (2012)
  - 'pertw_sch': variation of angular velocity proportional to Ω^3 (?).
  - 'pertw_loc': Loss of AM according to a local law: $\dot{J}_{\rm W} = TR_\star^2\Omega_\star^2$, where T is a constant.
  - 'pertw_ptm': loss of AM from mass-loss.
- P_PERTW (real): parameter of angular momentum loss attached respectively to each of the previous routines.
- LIM_JPZ (logical): If true, assumes $\mathrm{d}\Omega/\mathrm{d}m = 0$, else, $U_2 = 0$ (U_2 is the velocity of the meridional circulation) at the transition RZ/CZ (Warning: is almost not used any more).
- MIXED_MODES (logical): Account for AM transported by mixed modes.
- INIT_ROT (logical): True if angular velocity profile is read from a file (warning: do not use).
- NOM_DES_ROT (character): Control of the generation of ASCII files of the angular momentum diffusion variables (see Sect. C.3).
  - 'no_des': No file formation.

  - 'end_evol': Formation of the file at the end of the evolution, the file name is mon_modele_coeff_rota.da

  - 'all_mod': Formation of the file at the end of each time step, the file name is nnnn-mon_modele_coeff_ro nnnn being the model number. All files thus created are kept.
  - 'end_mod': Formation of the file at the end of each time step, the file name is mon_modele_coeff_rota.da only the last file created being available.
  - 'all_iter': Formation of the file at the end of each iteration of the resolution of the system of equations of angular momentum diffusion; files for debugging purposes.

- TAU_DISK (real): Lifetime of the disk (in Myrs).
- P_DISK (real): Rotation period of the disk (in days).
- W_SAT (real): Angular velocity at which the dynamo saturate.

3.11 NL_ETAT namelist members

- NOM_ETAT (character): name of the equation of state routine:
  - 'etat_ceff': equation of state by Eggleton et al. (1973), with Coulomb corrections (*calls 'etat_eff' in case of difficulty*).
  - 'etat_eff': equation of state by Eggleton et al. (1973), (*calls 'etat_gong2' in case of difficulty*).
  - 'etat_gong1': equation of state by GONG1 according to Christensen-Dalsgaard (1988), only H and He are taken into account and assumed to be fully ionized.
  - 'etat_gong2': equation of state by GONG2 according to Christensen-Dalsgaard (1988), only H and He4 are taken into account, degeneracy ignored.
  - 'etat_mhd': equation of state by Mihalas et al. (1988) (*calls 'etat_eff' in case of difficulty*) uses binary tables mhd1.bin... mhd7.bin built in the SUN_STAR_DATA subdirectory (see Sect. **??**).
  - 'etat_opal5Z': equation of state by OPAL 2005, (*calls 'etat_eff' in case of difficulty*) uses binary tables eos_opalZ*.bin built for *fixed Z* in the SUN_STAR_DATA subdirectory (see Sect. **??**).
  - 'etat_saha': (Baturin et al., 2013)
- F_EOS (array of characters): names of the equation of state files (up to 8 different).

3.12 NL_OPA namelist members

- NOM_OPA (character): name of the routine for calculating the Rosseland mean opacities:
  - 'opa_gong': simplified opacities (improved Kramers) according to Christensen-Dalsgaard (1988).
  - 'opa_houdek9': Houdek opacities version 9, (OPAL+Alexander), rational B-spline interpolation; uses binary tables built in the SUN_STAR_DATA/V9 subdirectory, see § **??** (Page **??**).
  - 'opa_int_zsx': OPAL93+Kurucz opacities with Yveline's adjustment, linear interpolations, very useful for testing.

  - 'opa_opal2_co', 'opa_opal2_cno': OPAL opacities with corrections for $Z > 0.1$ (see Sect. ??).
  - 'opa_opalCO': OPAL opacities with corrections for $Z > 0.1$ only for C and O.
  - 'opa_yveline': OPAL+Alexander opacities with Yveline's interpolation and adjustment; uses binary tables opa_yveline*.bin built in the SUN_STAR_DATA subdirectory, see § ?? (Page ??).
  - 'opa_yveline_lisse': OPAL+Alexander opacities with Yveline's adjustment, linear interpolation or smoothing; uses binary tables opa_yveline*.bin built in the SUN_STAR_DATA subdirectory (see Sect. ??).

- F_OPA (array of characters): opacity file names (up to 8).

Above a temperature higher than 70 10^6K, the matter is fully ionized, Cesam2k20 simplifies the opacity calculation by using the free-free formalism of Kramers according to Cox & Giuli (1968).

3.13 NL_NUC namelist members

- NOM_NUC (character): name of the routine for calculating the rates of thermonuclear reactions and initializing the chemical composition:
  - 'pp1': simplified calculation of the PP cycle (does not allow for microscopic diffusion). Elements taken into account: ^{1}H.
  - 'pp3': PP reactions, 3 elements with ^{2}H, ^{7}Li, ^{7}Be in equilibrium; intended for debugging, allows for microscopic diffusion, tabulation range: [1MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He.
  - 'ppcno9': PP+CNO reactions, 9 elements, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 40MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
  - 'ppcno9grad': PP+CNO reactions, 9 elements + 9 elements for the diffusion, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 40MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{20}Ne, ^{23}Na, ^{24}Mg, ^{27}Al, ^{28}Si, ^{31}P, ^{32}S, ^{40}Ca, ^{56}Fe.
  - 'ppcno9Fe': PP+CNO reactions, 9 elements+Fe, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{56}Fe.
  - 'ppcno10': PP+CNO reactions, 10 elements, ^{2}H, ^{7}Be in equilibrium, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
  - 'ppcno10Fe': PP+CNO reactions, 10 elements+Fe, ^{2}H, ^{7}Be in equilibrium, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{56}Fe.
  - 'ppcno10K': PP+CNO reactions, 10 elements+K, ^{2}H, ^{7}Be in equilibrium, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
  - 'ppcno10BeBFe': PP+CNO reactions, 10 elements + ^{6}Li, ^{9}Be, ^{11}B, ^{56}Fe with ^{2}H and ^{7}Be in equilibrium, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{56}Fe, ^{6}Li, ^{9}Be, ^{11}B.
  - 'ppcno11': PP+CNO reactions, 11 elements, ^{7}Be in equilibrium, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
  - 'ppcno12': PP+CNO reactions, 12 elements, tabulation range: [1MK, 80MK]. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{7}Be, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
  - 'ppcno12Be': PP+CNO reactions, 12 elements + ^{9}Be, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{7}Be, ^{9}Be, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.

- 'ppcno12Li': PP+CNO reactions, 12 elements + ^{6}Li, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{7}Be, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
- 'ppcno12BeBFe': PP+CNO reactions, 12 elements + ^{6}Li, ^{9}Be, ^{11}B, ^{56}Fe, tabulation range: [0.5MK, 80MK]. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{7}Be, ^{9}Be, ^{11}B, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{56}Fe.
- 'ppcno13gradNi': PP+CNO reactions, 25 elements. Elements taken into account: ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{7}Be, ^{9}Be, ^{11}B, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{20}Ne, ^{23}Na, ^{24}Mg, ^{27}Al, ^{28}Si, ^{31}P, ^{32}S, ^{40}Ca, ^{56}Fe, ^{58}Ni.
- 'ppcno3a9': PP+CNO+3α reactions, 9 elements, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 80MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O.
- 'ppcno3a12Ne': PP+CNO+3α+carbon reactions, 12 elements including ^{20}Ne, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 800MK]. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{18}O, ^{20}Ne, ^{2}H, ^{7}Li, ^{7}Be
- 'ppcno3aco': PP+CNO+3α+carbon+oxygen reactions, 17 elements, ^{2}H, ^{7}Li, ^{7}Be in equilibrium, tabulation range: [1MK, 3GK]. Elements taken into account:
- 'nrj31_YL': PP+CNO+3α+carbon+oxygen reactions. Elements taken into account: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, ^{18}O, ^{20}Ne, ^{19}F, ^{22}Ne, ^{24}Mg, ^{25}Mg, ^{26}Mg, ^{23}Na, ^{27}Al, ^{28}Si, ^{31}P, ^{32}S, neutrons.

The tabulation range is set according to the efficiency zone of the nuclear network chosen.

- nom_nuc_cpl (character): name of the compilation of thermonuclear reactions:
  - 'Cau-Fow': Compilation by Caughlan & Fowler (1988).
  - 'Adelb': Compilation by Adelberger et al. (1998).
  - 'Adelb_LUNA': Same as above, with rate from LUNA pour ^{14}N (Broggini et al., 2018).
  - 'NACRE': Compilation by NACRE by Aikawa et al. (2006).
  - 'NACRE_LUNA': Same as above, with rate from LUNA pour ^{14}N (Broggini et al., 2018).
  - 'NACRE2': Compilation by NACREII (Xu et al., 2013).
  - 'Reaclib': Compilation by Reaclib (Cyburt et al., 2010).
- MITLER (logical) : screening effect according to Mitler (1977).

3.14 NL_ATM namelist members

- NOM_ATM (character): name of the routine for atmosphere retrieval:
  - 'lim_atm': atmosphere retrieval from a $T(\tau)$ law.
  - 'lim_gong1': simplified atmosphere retrieval, GONG1 case.
  - 'lim_tau1': simplified atmosphere retrieval, single-layer atmosphere.
- NOM_TDETAU (character): name of the $T(\tau)$ law used for atmosphere retrieval:
  - 'edding': Eddington's fully radiative $T(\tau)$ law.
  - 'hopf': Hopf's fully radiative $T(\tau)$ law.
  - 'hsra': HSRA's empirical, not fully radiative $T(\tau)$ law.
  - 'K5750': non-fully radiative $T(\tau)$ law, derived from Kurucz's solar atmosphere model, $T_{\rm eff} = 5750$ K.

- `'K5777'`: non-fully radiative $T(\tau)$ law, derived from Kurucz's solar atmosphere model, $T_{\rm eff} = 5777$ K.
- `'roger'`: non-fully radiative $T(\tau)$ law, derived from Kurucz's atmosphere models, with metallicity $[{\rm Fe/H}] \in \{+0.2, 0.0, -0.5, -1.0\}$. Last one includes "enhanced" alpha elements.
- `'kurucz'`: Modified version of `'K5750'` that can read any Kurucz atmosphere model.
- `'verna'`: $T(\tau)$ relation of Vernazza et al. (1981), fit to reproduce solar chromosphere observational data.
- `'krisw'`: $T(\tau)$ relation of Krishna Swamy (1966)
- `'tramp'`: Interpolation of $T(\tau)$ relation in table provided by Trampedach et al. (2014). **Do not use: implementation in progress**.
- `'ball21'`: $T(\tau)$ relation obtained using functional fit of Ball (2021), calibrated on Trampedach et al. (2014)'s solar 3D atmosphere model.

- `TAU_MAX (real)`: optical depth at the envelope boundar.
- `LIM_RO (logical)`: external boundary condition in density, otherwise in pressure.

3.15 Advanced setup of the chemical compositions

Although the executable is built once and for all, it is possible to externally adapt a certain number of parameters. These adaptations are made, upon request, through specific files to be placed in the sub-directory where the calculation takes place. Only models calculated in this environment will be affected. In general, in the concerned directory, customization can affect either a specific model or all models to be calculated in this directory. Cesam2k20 first searches if customization exists for the model to be calculated, then, if not, if it exists for all models in the directory. Examples of such customization files can be found in the `exploit` sub-directory of the distribution. Thus, you can:

- Use a different initial mixture from those implemented.
- Modify abundance reports of a mixture.
- Redefine isotopic ratios.

For safety, as explicitly stated for each particular case, some of the customization files concerning the chemical composition will only be taken into account if the parameter `modif_chim` of the NAMELIST `nl_chim` in the data file is `.true.` (see Sect. 3.7).

3.15.1 Mixture

To use a different initial mixture[6] from those implemented, code `nom_abon='mixture'` in the data file and, in the operating environment, place a file named `mixture` containing the desired mixture in Dex. Cesam2k20 *rejects* any mixture whose normalization is different from $H_1 = 12$. The following example is found in the `exploit` sub-directory:

```
1 &NL_MIXTURE
2  ab(1)=12.00d0,  ab(2)=10.d0,   ab(3)=2.69d0,  ab(4)=2.15d0,  ab(5)=2.60d0
3  ab(6)=4.55d0,   ab(7)=3.97d0,  ab(8)=4.87d0,  ab(9)=1.56d0,  ab(10)=4.08d0
4  ab(11)=2.33d0,  ab(12)=3.58d0, ab(13)=2.47d0, ab(14)=3.55d0, ab(15)=1.45d0
5  ab(16)=3.21d0,  ab(17)=1.5d0,  ab(18)=2.52d0, ab(19)=1.12d0, ab(20)=2.36d0
```

[6] See also Sect. ??.

```
 ab(21)=-1.17d0, ab(22)=1.02d0, ab(23)=0.d0,   ab(24)=1.67d0, ab(25)=1.39d0
 ab(26)=3.5d0,   ab(27)=0.92d0, ab(28)=2.25d0
/
```

In the abon_ini routine, the order and identification of elements are given by their indices, in fact, those of the OPAL opacities. Therefore, *all* models calculated in this directory, if they use the nom_abon='mixture' option, will have initial abundances consistent with those in the mixture file. Another possibility allows for further customization; it involves giving the mixture file the generic name of the model with the .mix extension, e.g., my_model.mix. From then on, *only* the my_model model will have the desired initial chemical composition. With the mixture option, Cesam2k20 *prioritizes* using the my_model.mix file; in its absence, the mixture file is used; the calculation is only performed if one of these two files exists in the environment. This option requires coding MODIF_CHIM=.true., see Sect. 3.7.

Problem: *With the* nom_abon='mixture' *option, the relative abundances of metals should, in principle, correspond to those of the opacities, or at least be supported by the opacity routine used.*

3.15.2 Isotopic ratios

To use isotopic ratios different from those that Cesam2k20 takes by default, place in the calculation environment a file named my_model.rap_iso specifying the isotopic ratios to use. If *all* models in the environment should be calculated with modified ratios, give the previous file the name rap_iso. For example:

```
&NL_RAP_ISO
 be7sbe9   = 1.d-25,
 be7sz     = 1.d-29,
 c13sc12   = 1.10d-2,
 h2sh1     = 3.01d-5,
 he3she4   = 1.1d-4,
 he3she4z  = 4.185d-4,
 li6sli7   = 7.5d-2,
 mg25smg24 = 0.0125,
 mg26smg25 = 0.013,
 ne22sne20 = 6.79d-2,
 n15sn14   = 0.366d-2,
 o17so16   = 0.038d-2
 o18so16   = 0.008d-2
/
```

The variable names used suggest the notations of the isotopic ratios be7sbe9: isotopic ratio ^{7}Be/^{9}Be. he3she4z is the isotopic ratio ^{3}He/^{4}He on the ZAMS when the initial deuterium has been transformed into ^{3}He, be7sz is the quasi-zero abundance of initial ^{7}Be in Z. For safety, this file will only be considered if the modif_chim parameter of the NAMELIST nl_chim in the data file is .true. (see Sect. 3.7).

3.15.3 Customization of abundance ratios

Once the mixture is defined, for example solaire_aag21 (Asplund et al., 2021), it is possible to modify the abundances. To do this, create a file named my_model.modif_mix in the calculation environment, in which the modifications in DeX to be made are indicated. If all models in the environment should

be calculated with these modifications, give the previous file the name modif_mix. Example found in the exploit directory:

```
&nl_modif_mix
 add_Li=2.d0;  add_Be=0.d0; add_B=0.d0
 add_C=-1.0d0; add_N=0.d0;  add_O=-0.3d0; add_F=0.d0;  add_Ne=0.d0
 add_Na=0.d0;  add_Mg=0.d0; add_Al=0.d0;  add_Si=0.d0; add_P=0.d0
 add_S=0.d0;   add_Cl=0.d0; add_Ar=0.d0;  add_K=0.d0;  add_Ca=0.d0
 add_Sc=0.d0;  add_Ti=0.d0; add_V=0.d0;   add_Cr=0.d0; add_Mn=0.d0
 add_Fe=0.d0;  add_Co=0.d0; add_Ni=0.d0;  add_Z=-1.d0
/
```

The notation suggests the quantities to be modified; thus add_Li=2.d0 is the number of dex to add to the lithium abundance. add_Z=-1.d0 means that the abundances *of all metals* will be decreased by one DeX. *For safety*, this file will only be considered if the modif_chim parameter of the NAMELIST nl_chim in the data file is .true. (see Sect. 3.3).

If garde_xish=.false. is coded in the data file (see Sect. 3.7.1), the add_Z parameter, which affects all metals as a whole, has no impact on the determination of the metal/Z ratios. In contrast, a non-zero value of add_C, for example, will affect the metal/Z ratios.

3.15.4 Customization of wind chemical composition

In the standard case, with the mdot data not being null (see Sect. 3.3), the chemical composition of the mass lost or gained by the wind is that of the outermost layer of the model. It is possible to customize this chemical composition so that it differs from that of the outer layer. To do this, place a file named my_model.vent in the calculation environment, in which the mass fractions of each chemical element in the mass input or output are indicated. If all models in the environment should be calculated with these modifications, give the previous file the name wind. An example is found in the EXPLOIT sub-directory:

```
&nl_wind
 vt_H=0.7347,     vt_He=0.2483,    vt_Li=1.033e-08, vt_Be=1.72816e-10,
 vt_B=4.859e-09,  vt_C=2.899e-03,  vt_N=8.493e-04,  vt_O=7.885e-03,
 vt_F=4.182e-07,  vt_Ne=1.768e-03, vt_Na=3.501e-05, vt_Mg=6.736e-04,
 vt_Al=6.078e-05, vt_Si=7.434e-04, vt_P=8.198e-06,  vt_S=3.704e-04,
 vt_Cl=4.924e-06, vt_Ar=7.315e-05, vt_K=3.845e-06,  vt_Ca=6.541e-05,
 vt_Sc=4.126e-08, vt_Ti=3.041e-06, vt_V=3.888e-07,  vt_Cr=1.856e-05,
 vt_Mn=1.357e-05, vt_Fe=1.287e-03, vt_Co=3.492e-06, vt_Ni=7.6092e-05
/
```

The notations are suggestive, so vt_Li=1.033e-08 is the mass fraction of lithium in the wind. A renormalization is performed when used, so it is not necessary to normalize the mass fractions in this file; some can be zero or even negative. Depending on the sign of MDOT, the amount of matter, added or subtracted, is assumed to come from the outer CZ, which always exists. The wind file can be constructed[7] by the wind_file.f program in the exploit sub-directory.

For safety, this file will only be considered if the modif_chim parameter of the NAMELIST nl_fdon in the data file is .true. (see Sect. 3.7).

[7] The chemical composition of the example is that of the meteoritic mixture of Grevesse & Sauval (1998). Furthermore, it is not necessary here to respect the 'case'.

3.15.5 Planetoid falls

During an evolution, Cesam2k20 allows simulating a fall of planetoids over a limited time interval. To do this, place a file named `mon_modele.planet` in the calculation environment, in which the mass fractions of each element of the chemical composition of the planetoids, the total number of planetoids that will be received by the star, the time interval concerned by the fall, and the intensity profile of the fall are indicated. If all models in the environment must be calculated with these modifications, name the previous file `planet`. An example is provided in the EXPLOIT sub-directory:

```
&nl_planet
 vt_H=0.7347,     vt_He=0.2483,       vt_Li=1.033e-08, vt_Be=1.72816e-10,
 vt_B=4.859e-09,  vt_C=2.899e-03,     vt_N=8.493e-04,  vt_O=7.885e-03,
 vt_F=4.182e-07,  vt_Ne=1.768e-03,    vt_Na=3.501e-05, vt_Mg=6.736e-04,
 vt_Al=6.078e-05, vt_Si=7.434e-04,    vt_P=8.198e-06,  vt_S=3.704e-04,
 vt_Cl=4.924e-06, vt_Ar=7.315e-05,    vt_K=3.845e-06,  vt_Ca=6.541e-05,
 vt_Sc=4.126e-08, vt_Ti=3.041e-06,    vt_V=3.888e-07,  vt_Cr=1.856e-05,
 vt_Mn=1.357e-05, vt_Fe=1.287e-03,    vt_Co=3.492e-06, vt_Ni=7.6092e-05,
 ypl=0.d0,        zpl=0.9d0,
 n_planet=10,     profil='rectangle', age_deb=5.d0,    age_fin=10.d0,
 r_giration=1.d0, t_giration=1.d0
/
```

The notations are suggestive. With `vt_H` and `vt_He` *non-null*, non-zero values of `ypl` and/or `zpl` allow modifying the mass fractions of hydrogen `X`, helium `Y`, and metals `Z` of the planetoids. With the previous `planet` file, the mass abundance of helium will be $Y = 0$, that of metals $Z = 0.9$, and that of hydrogen $X = 1.0 - 0.0 - 0.9 = 0.1$. `age_deb` and `age_fin` are respectively the ages, in million years, of the beginning and end of the fall. `r_giration` is the gyration radius in AU, and `t_giration` is the gyration time in years. Cesam2k20 estimates the angular velocity of the planetoids by assuming that they describe an arc of $\pi/2$ during a gyration time. A negative value of the gyration time corresponds to a negative input of angular momentum, i.e., retrograde. With a zero value of the gyration time, there is no input of angular momentum.

In the `planetoides.f90` routine, Cesam2k20 proposes 4 profiles:

1. `rectangle`: The intensity of the fall is constant.
2. `triangle`: The intensity profile of the fall is triangular, with the maximum centered in the middle of the time interval.
3. `parabole`: The intensity profile of the fall is parabolic, with the maximum centered in the middle of the time interval.
4. `gauss`: The intensity profile of the fall is a Gaussian limited to the time interval. The maximum is centered in the middle of the time interval. The standard deviation is arbitrarily taken equal to one-third of the time interval.

This option requires coding `modif_chim=.true.`, see 3.7.

3.16 Numerical settings

The numerical settings, defined in the `set_num_params.f90` routine depending on the `PRECISION` parameter of the `mon_modele.don` data file, can be customized in an additional namelist `NL_RLG`. E.g.

```
&NL_RLG
num%m_qs=2,
num%m_ch=3,
num%m_rot=2,
num%m_tds=2,
num%m_ptm=2,
num%order=1,
num%precix=1.d-4,
num%precit=0.1d0,
/
```

The meanings of the parameters are described in details in Sect. 7.5.2.

Adjusting the distribution constants can be delicate. The default values are `ctel=0`, `ctep=-1`, `ctem=15`, `cter=0`, `ctet=0`. Be careful that `ctep` and `ctet` must be negative, as they respectively affect pressure and temperature, quantities decreasing from the center to the surface.

As with initial abundances (see Sect. 3.15.1), it is possible to further customize these settings by giving the settings file the same name as the model with the `.rg` extension, for example: `mon_modele.rg`. If it exists in the environment, the parameters from this file will be considered for the model named `mon_modele`. In its absence, the quantities from the `reglages` file will be used; if it does not exist, the calculation will not be performed.

Table 3.1: Values of the numerical parameters according to the various precision options provided.

	'pr'	'er'	'sp'	'sr'	'av'	'co'	'pl'	'mj'
integer:								
ini0	4		5	5		6	6	6
l0	0					0	0	5
m_qs	2	1		1	1			
m_ch	2					3		3
m_vth	3							
m_rot	3	1		1				
m_tds	2							
m_ptm	2							
n_atm	75		100	100			200	100
n_max	2500						4000	
n_min	150						3000	
ordre	2							
yld_rep	1							
yld_osc	1							
m_qs2d	3							
m_rot2d	3							
m_legendre	2							
n_theta	100							
fast_rot	-1							
real:								
ctel	0.0							
ctep	-1.0							
ctem	15.0	0.0		0.0				
cter	0.0	15.0		15.0				
ctet	-1.0							
d_grav	$5 \cdot 10^3$		0.5	0.5		0.5	0.5	0.5
dlntc	0.07		0.05	0.05		0.05	0.05	0.05
drhoc	-1.0						0.05	
dteff	-1.0						0.005	
dlogg	-1.0						0.1	
dalpha	0.03							
dlum	-1.0						0.03	
dsenv	$5 \cdot 10^{-4}$							
evolved	0.1							
dn_fixe	0.05							
dpsi	0.05							

dt0	1.0							
dtmax	200.0		50.0	50.0		50.0	10.0	50.0
dtmin	$1 \cdot 10^{-15}$							
fmin_abon	0.05		0.01			0.01	0.01	0.01
loc_zc	$1 \cdot 10^{-3}$		$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$		$1 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-4}$
precix	$1 \cdot 10^{-3}$		$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$		$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-4}$
precit	0.15		0.05	0.05		0.05	0.01	0.05
psi0	0.08		0.06	0.06		0.06	0.03	0.03
ro_test	0.1							
q0	0.05				0.0	0.0	0.05	0.01
yld_dx	-1.0							
yld_rac	1.0							
yld_atm	$1 \cdot 10^{-4}$							
dt_m	-2.5							
x_tams	0.01							
dx_tams	$1 \cdot 10^{-4}$							
logical:								
ajuste	T				F			
en_masse	T	F		F				
kipp	F				T			
lisse	F				T			
mu_saha	T				F			
mvt_dis	T							
new_bv	F							
yld_evol	F							
yld_hr	F							
yld_l23	F							
d2	T							
general	F							
simple	T							
extended_osc2d	F							
vdiff_out	F							
grad_out	F							
th_out	F							
eos_inplace	F							
ec	F							
character:								
ltau_rep	'cub'							
ec_s	'average'							

	'lm'	'sa'	'hp'	'np'	'nr'	'mx'
integer:						
ini0	5	5	6	2	3	5
l0	0	5	7		0	4
m_qs			3	1	1	
m_ch			4	2		
m_vth						
m_rot				1		
m_tds						
m_ptm						
n_atm		100	150	50	50	100
n_max						
n_min						
ordre				1	1	
yld_rep						
yld_osc						
m_qs2d						
m_rot2d						
m_legendre						
n_theta						
fast_rot						
real:						
ctel						
ctep						
ctem					0.0	
cter					15.0	
ctet						
d_grav		0.5	0.05	0.5	$5 \cdot 10^3$	0.5
dlntc		0.05	0.03	0.1	0.1	
drhoc						
dteff						
dlogg						

dalpha						
dlum						
dsenv						
evolved						
dn_fixe	0.1					
dpsi						
dt0						
dtmax	300.0	50.0	50.0	300.0	300.0	50.0
dtmin						
fmin_abon		0.01	0.01	0.05	0.05	0.01
loc_zc		$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-5}$
precix		$1 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-5}$
precit	0.2	0.02	0.01	0.3	0.3	0.02
psi0		0.06	0.05	0.1	0.1	$1 \cdot 10^{-8}$
ro_test			0.05			
q0	0.0	0.05	0.01	0.0	0.0	
yld_dx						
yld_rac						
yld_atm						
dt_m						
x_tams						
dx_tams						
logical:						
ajuste				F	F	
en_masse					F	
kipp	T			T	T	
lisse						
mu_saha	F			F	F	
mvt_dis				F	F	
new_bv						
yld_evol						
yld_hr						
yld_l23						
d2						
general						
simple						
extended_osc2d						
vdiff_out						
grad_out						
th_out						
eos_inplace						
ec						
character:						
ltau_rep						
ec_s						

Chapter 4

For developers

Contents

4.1 Adding a thermonuclear reaction chain

To implement a new thermonuclear reactions chain, it is often necessary to add one or more isotopes, as well as one or more thermonuclear reactions.

o add an isotope [1]:

1. In the `mod_nuc` module, increase the `parameter niso_tot` by one unit, for example: `niso_tot=28`.
2. In the `taux_nuc` routine:
   (a) In the declarations, add the excess mass of the isotope and its symbol, for example: `Fe56=-60.6054d0`.
   (b) Add the mass of the isotope (it is advisable to add the mass of the isotope in the physical constants routines `ctes_85`, `ctes_94`, etc.), and consequently in the declarations of the `mod_donnees` module and in the `only` restriction statement of the `taux_nuc` routine).
   (c) Add the charge of the isotope, for example: `zit(28)=11`.
   (d) Add the name of the isotope, for example: `'B11 '`. (these 4-character strings are left-justified except for 2-character symbols which are centered.).
3. If necessary, in the `abon_ini` routine:
   (a) Add an isotopic ratio for this new isotope:
      i. Enter its value in the list arranged at the end of the routine.
      ii. Indicate its name in the NAMELISTs `nl_rap_iso` and `nl_modif_mix` in the declarative part of the routine.
4. If necessary, in the `saha` routine, add the ionization potentials for this new element.

o add a reaction :

1. In the `taux_nuc` routine:
   (a) Complete the comment list by indicating the symbol of the reaction and its index, for example: `reaction 46 : B11(p,g)C12`.

[1] Cesam2k20 allows the use of chemical elements from hydrogen ($Z = 1$) up to nickel ($Z = 28$). To introduce elements of higher mass, place them in the `abon_ini` routine applying a procedure similar to that described for adding an isotope.

(b) Initialize the name of the reaction, the mass defect, and the charges of the nuclei involved, for example:

```
nom_react(2) = 'H2(p,g)He3'
nuc          = H2+p-He3
qt(2)        = nuc
izzt(2,1)    = 1
izzt(2,2)    = 1
```

(c) Enter the rate of the reaction[2] `rt(i)` in ln and do not forget to include the 1! or 2! or 3! from the denominator, for example: `reaction 8 : C12(p,g)N13(e+ nu)C13 z0=6, z1=1` recalculating, if necessary, the coefficients from $S(0)$, $S'(0)$, $S"(0)$ (**?**, eq. 4-49).

2. Create the thermonuclear reaction routine by following the example of existing routines, such as `ppcno12BeBFe.f90`.
3. In the `tabul_nuc` routine:
   (a) Add the new chain, for example: `case('ppcno12')`.
   (b) Enter the parameters for the rate calculation, the number of reactions, the index correspondences, the isotopes used, etc., for example:

```
nreac=30
ind(16)=31
WRITE(2,6)
WRITE(*,6)
```

4. In the `mod_nuc` module:
   (a) Increase the `parameter nreac_tot` by one unit, for example: `nreac_tot=46`
   (b) Include the name of the reaction chain routine using `include`, for example: `include 'ppcno9.f90'`
5. In the generic routine `nuc`:
   (a) Call the new routine.
6. In the `src/Makefile`: add the routine in the `mod_nuc.o` section
7. Test the modified algorithms using the programs in the `tests` sub-directory. Create your own test if needed.

[2] It is for historical reasons that these calculations are done in ln. The reasons were to avoid, on one hand, truncation errors and on the other hand, capacity overflows with the limitation to 10^{38}.

Chapter 5

Debug

Contents

5.1 Debugging

The modular structure of Cesam2k20 allows recompiling only the modified module to which the routine containing the bug belongs. During installation, it is useful to create, along with the operating library `libcesam2k.a`, a debug library `libcesam2k-dbg.a`. This library is automatically created when running `make debug`.

La structure de Cesam2k20 en modules permet une vérification efficace de la syntaxe par le compilateur. Le revers de la médaille est que la mise au point et/ou le debug nécessitent de recompiler tout le module auquel appartient la routine dans laquelle le bug est recherché. Lors de l'installation, il est utile de créer, conjointement à la bibliothèque d'exploitation `libcesam2k.a` une bibliothèque de debug `libcesam2k-dbg.a`. Cette bibliothèque est créée automatiquement lors de l'exécution du script `genere_cesam2k-dbg` du sous-directory `SCRIPTS`. Pour ce faire, utiliser les options de debug du compilateur; la procédure `compile2k-dbg` du sous-directory `SCRIPTS` en est un exemple. Le temps nécessaire à la construction de la bibliothèque de debug est significativement plus court que celui de la construction de la bibliothèque d'exploitation, facteur 5 à 10 par exemple, facteur qui se retrouve, mais dans l'autre sens, lors d'une exécution.

Exemple: un bug a été détecté dans la routine `des_m` du module `mod_cesam`[1]. On introduit des instructions de debug dans `des_m`. Dans le sous-directory `TESTS` on exécute le programme `test_cesam` dans lequel on inclut le module `mod_cesam` :

```
1
2       INCLUDE '../SOURCE/mod_cesam.f'
3
4   c*******************************************************************
5
6       PROGRAM test_cesam
7
8       USE mod_cesam
```

[1]Les *bons* compilateurs indiquent le nom des routines où ils détectent un bug.

```
9
10      IMPLICIT NONE
11
12      CALL cesam
13
14      STOP
15
16      END PROGRAM test_cesam
17
```

Au link, le fichier `mod_cesam.mod` créé dans le sous-directory `TESTS` sera utilisé et non pas celui du sous-directory `SOURCE`. Lors de l'exécution, les instructions de debug, alors prises en compte, permettront de détecter le défaut. Une fois la routine `des_m` corrigée, dans le sous-directory `SOURCE`, l'exécution du script: `mod_repl mod_cesam`, remplacera le module `mod_cesam` dans la bibliothèque et recréera l'exécutable `cesam2k.out`. Ainsi, aucun fichier du sous-directory `SOURCE` n'a été déplacé.

5.2 Known Bugs

1. Poor convergence, the iterative process "gets stuck": convergence begins, then the corrections no longer decrease, or oscillate around a fixed value. *Possible cause*: non-differentiable interpolation of opacity, e.g., with linearly interpolated opacities. *Remedy*: if increasing the order of opacity interpolation is the theoretical solution, it is often impractical, and there is a risk of oscillations due to interpolation steps that are too large, or not smooth enough transitions between tables; practically, one must settle for less precision to pass the difficult point. This type of difficulty often occurs in the atmosphere due to the transition between opacity tables of different origins. Under certain conditions, Cesam2k20 forces convergence.

2. Poor functioning of the control over the temporal variation of specific entropy $T\mathrm{d}S$, resulting in the message: `"TdS varies too much... decreasing the time step"`, the time step decreases without solving the problem. *Possible cause*: temporal discontinuity of the chemical composition with a convective core. *Remedy*: sometimes, although it seems paradoxical and the reason is not understood, simply restarting the evolution with a larger time step can resolve the issue; one can also attempt to modify `d_grav`, the value of the parameter controlling the allowed variation of TdS; increasing its value reduces precision but increases robustness; one can eliminate the control by assigning a very large value (10^{10}) to this control parameter; it is initialized in the `cesam` routine, *cf.* §7.7.3 (Page 158). Occasionally, the difficulty can be circumvented by using the Kippenhahn approximation, *cf.* §6.4.1 (Page 72). To adjust these parameters, it is practical to use the possibility to customize the settings of Cesam2k20, *cf.* §3.15 (Page 30).

3. Divergence at the first time step. *Possible cause*: initial time step too high. *Remedy*: decrease the initial `dt0` initialized in the `cesam` routine according to the required precision type. To adjust these parameters, it is practical to use the possibility to customize the settings of Cesam2k20, *cf.* §3.15 (Page 30).

4. The localization of zones radiatives / zones convectives limits on a grid point does not work. *Possible cause*: the ∇_{rad} and ∇_{ad} gradients do not intersect clearly, or the limit moves significantly, i.e., when a convective core disappears. *Remedy*: when the gradients are close, the RZ/CZ limit is poorly defined, trying to force a grid point exactly is illusory and unnecessary, Cesam2k20 adapts to the situation by imposing the position obtained after the number of iterations set in the routine `cesam` according to the type of required precision. To adjust these

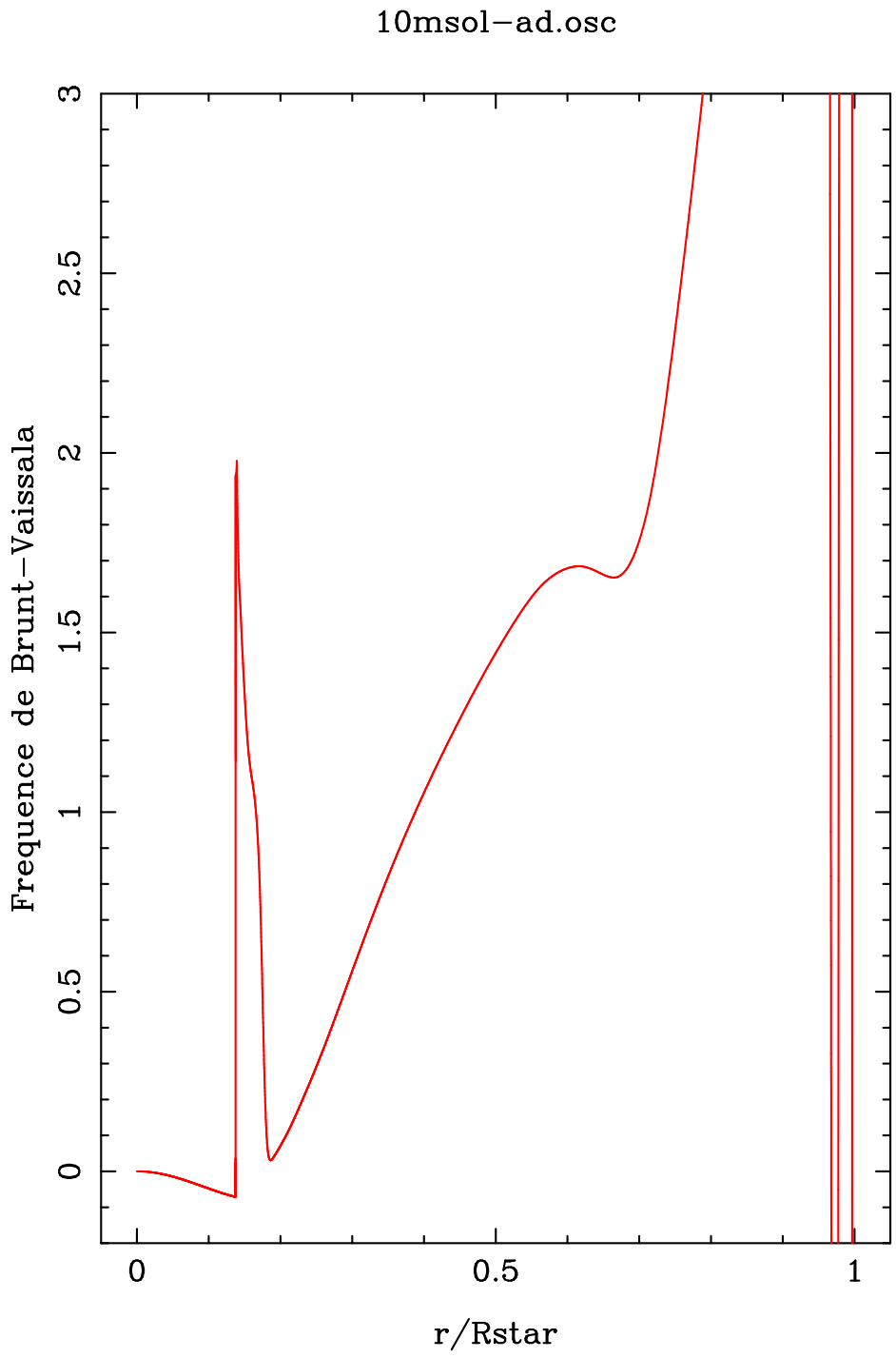


Figure 5.1: Profil de la fréquence de **v** d'un modèle de $10M_{\odot}$ de séquence principale ($X_c = 0.35$), calculé avec la prodédure décrite §12 (Page 44).

parameters, it is practical to use the possibility to customize the settings of Cesam2k20, *cf.* §3.15 (Page 30).

5. Inability to construct a ZAMS model. *Possible cause*: use of a chemical composition too far from equilibrium, e.g., the one used for PMS models with cosmic abundances of H2 and Li7. *Remedy*: adjust the initial chemical composition. To adjust these parameters, it is practical to use the possibility to customize the initial abundances of Cesam2k20, *cf.* §3.15 (Page 30).

6. For high masses $\sim 10 M_\odot$ and above, after the disappearance of the central hydrogen, zones radiatives / zones convectives appear and disappear, hindering convergence. *Possible cause*: presence of semi-convective zones. *Remedy*: chronic difficulty, Cesam2k20 does not provide a treatment for semi-convection.

7. Impossible convergence using the corrective term of turbulent pressure $\frac{\mathrm{d}\ln P_{\mathrm{gaz}}}{\mathrm{d}\ln P}$, i.e., with a strictly negative value for the argument `cpturb` of the NAMELIST `NL_CONV`. *Possible cause*: This problem likely results from an inconsistency of the MLT. *Remedy*: none in the current state.

8. The compiler refuses to compile a routine with a high level of optimization. *Possible cause*: Compiler whim. *Remedy*: use a lower compilation level.

9. Very poor convergence of the atmosphere calculation. *Possible cause*: passing through a sharp point in the opacity table, or modification of the grid index of the $n_\star$ index point. *Remedy*: often, after a few time steps, one moves away from the sharp points, and these difficulties disappear. Use smooth interpolation functions for opacity tables.

10. Very poor convergence of the atmosphere calculation. *Possible cause*: pressure boundary condition that cannot be satisfied at the high precision level required for atmosphere integration. *Remedy*: use the density boundary condition.

11. Poor convergence of the temporal integration algorithm for ^{2}H, occurs towards the end of the pré-séquence principale, sometimes jointly with opacity table outputs. *Possible cause*: oscillations when reaching the equilibrium of deuterium. *Remedy*: Cesam2k20 "forces convergence", which, often at the level of 10^{-5}, is not bad.

12. Violent oscillations of the quantity $A = \frac{1}{\Gamma_1}\frac{\partial \ln P}{\partial \ln R} - \frac{\partial \ln \rho}{\partial \ln R}$ in a `mon_modele-*.osc` file *cf.* §C.1 (Page 345). These oscillations have several described causes, *cf.* §6.9.9 (Page 121). Although much effort has been devoted to this difficulty, so far the least bad remedy is to manually remove the incriminated layers. Diffusion of chemical elements largely avoids this phenomenon. When it is not required and a good representation of the $\mathbf{v}$ frequency is needed, in the data file, *cf.* §3.3 (Page 14), code `'sa'` for the precision type, and introduce some turbulent diffusion:

```
/
&NL_DIFF
DIFFUSION=.TRUE.,
NOM_DIFFM='diffm_0',
NOM_DIFFT='difft_nu',
D_TURB=10.d0,
RE_NU=1.d0,
NOM_FRAD='no_frad'
/
```

which will eliminate discontinuities in the first derivative of the chemical composition. The cost is a significantly higher computational effort than with `DIFFUSION=.FALSE.`. La figure 5.1

(Page 41) represents the **v** frequency profile obtained for a $10M_\odot$ main sequence model ($X_c = 0.35$), calculated with the procedure just described. During the evolution, the convective core regressed from $0.22R_\star$ to $0.14R_\star$.

13. Poor convergence for atmosphere retrieval. This problem sometimes occurs when resuming a model. The difficulty arises because the atmosphere that was resumed in binary is not sufficiently close to the one that Cesam2k20 wants to calculate. *Remedy*: delete the binary atmosphere model `mon_modele.atm`. To initialize, Cesam2k20 uses the standard atmosphere.

14. Poor convergence, time step tending towards 0, and/or oscillations of the angular velocity near zones radiatives / zones convectives limits. The causes are not identified. A consequence is the generation of chemical composition instabilities, which, transmitted to density, generate very noisy coefficients for the angular momentum diffusion equations. Increasing the turbulent diffusion coefficient `d_turb`, *cf.* §?? (Page ??), sometimes helps to eliminate anomalies.

15. Very noisy profile of the rate of change of mean molecular weight Λ, *cf.* §6.8.5 (Page 102), near zones radiatives / zones convectives limits. Unknown remedy. This anomaly generally has no direct consequence on the profile of the angular velocity.

5.3 Bugs connus

1. Mauvaise convergence, le processus itératif "tourne en rond": la convergence s'amorce, puis les corrections ne diminuent plus, ou oscillent autour d'une valeur fixe. *Cause possible*: interpolation non dérivable de l'opacité *e.g.* avec des opacités interpolées linéairement. *Remède* : si l'augmentation de l'ordre d'interpolation des opacités est la solution théorique, elle est souvent impraticable, et il y a risque d'oscillations à cause de pas d'interpolation trop grands, ou de raccords pas assez lisses entre tables; pratiquement, il faut se contenter d'une précision moins grande pour passer le point difficile. Ce type de difficulté se produit assez souvent dans l'atmosphère, à cause du raccord entre tables d'opacités d'origines différentes. Dans certaines conditions, Cesam2k20 *"force la convergence"*.

2. Mauvais fonctionnement du contrôle de la variation temporelle de l'entropie spécifique (TdS), on obtient le message: "TdS varie trop... diminution du pas temporel", le pas temporel diminue sans que, pour autant, le problème disparaisse. ***Cause possible***: discontinuité temporelle de la composition chimique avec un cœur convectif. ***Remède*** : parfois, bien que celà semble paradoxal et que l'on n'en comprenne pas la raison, il suffit de reprendre l'évolution avec un pas temporel plus grand pour que le point délicat soit passé; on peut aussi tenter de modifier `d_grav`, valeur du paramètre de contrôle de la variation admise du TdS; augmenter sa valeur, diminue la précision mais augmente la robustesse; on peut éliminer le contrôle en donnant une valeur très grande (10^{10} à ce paramètre de contrôle; il est initialisé dans la routine `cesam`, *cf.* §7.7.3 (Page 158). On contourne parfois la difficulté en utilisant l'approximation de Kippenhahn *cf.* §6.4.1 (Page 72). Pour ajuster ces paramètres il est pratique d'utiliser la possibilité de personnaliser les réglages de Cesam2k20 *cf.* §3.15 (Page 30).

3. Divergence au premier pas temporel. ***Cause possible***: pas temporel trop élevé. ***Remède***: diminuer le `dt0` initialisé dans la routine `cesam` suivant le type de précision requis. Pour ajuster ces paramètres il est pratique d'utiliser la possibilité de personnaliser les réglages de Cesam2k20 *cf.* §3.15 (Page 30).

4. La localisation des limites zones radiatives / zones convectives sur un point de grille ne fonctionne pas. ***Cause possible***: les gradients $\nabla_{\rm rad}$ et $\nabla_{\rm ad}$ ne se croisent pas franchement, ou encore la limite se déplace notablement *i.e.* , lors de la disparition d'un cœur convectif. ***Remède***: quand

les gradients sont voisins, la limite RZ/CZ est mal définie, y vouloir forcer un point de grille exactement est illusoire et inutile, Cesam2k20 s'accomode de la situation en imposant la position obtenue après le nombre d'itérations fixé dans la routine `cesam` suivant le type de précision requis. Pour ajuster ces paramètres, il est pratique d'utiliser la possibilité de personnaliser les réglages de Cesam2k20 *cf.* §3.15 (Page 30).

5. Impossibilité de construire un modèle de séquence principale d'âge zéro. ***Cause possible***: utilisation d'une composition chimique trop éloignée de l'équilibre *e.g.* celle utilisée pour les modèles de PMS avec les abondances cosmiques de H2 et Li7. ***Remède*** : adapter la composition chimique initiale. Pour ajuster ces paramètres, il est pratique d'utiliser la possibilité de personnaliser les abondances initiales de Cesam2k20, *cf.* §3.15 (Page 30).

6. Pour des masses élevées $\sim 10 M_\odot$ et plus, après la disparition de l'hydrogène central, des ZC apparaissent, disparaissent nuisant à la convergence. ***Cause possible*** : présence de zones semi-convectives. ***Remède***: difficulté chronique, Cesam2k20 ne prévoit pas de traitement de la semi-convection.

7. Convergence impossible en utilisant le terme correctif de pression turbulente $\frac{\mathrm{d}\ln P_{\mathrm{gaz}}}{\mathrm{d}\ln P}$ *i.e.* avec une valeur strictement négative pour l'argument `cpturb` de la NAMELIST `NL_CONV`. ***Cause possible***: Ce problème résulte vraisemblablement d'une incohérence de la MLT. ***Remède*** : néant en l'état actuel.

8. Le compilateur refuse de compiler une routine avec un niveau élevé d'optimisation. ***Cause possible***: Caprice du compilateur. ***Remède***: utiliser un niveau de compilation moins élevé.

9. Très mauvaise convergence du calcul d'atmosphère. ***Cause possible***: passage d'un point anguleux de la table d'opacité, ou modification de l'indice de grille du point d'indice $n_\star$. ***Remède***: souvent, après quelques pas temporels, on s'écarte des points anguleux, et ces difficultés disparaissent. Utiliser des fonctions d'interpolation dérivables pour les tables d'opacité.

10. Très mauvaise convergence du calcul d'atmosphère. ***Cause possible***: condition limite sur la pression qui ne peut être satisfaite au niveau de pécision élevé requis par l'intégration de l'atmosphère. ***Remède***: utiliser la condition limite sur la densité.

11. Mauvaise convergence de l'algorithme d'intégration temporelle pour ^{2}H, se produit vers la fin de la pré-séquence principale, se produit parfois conjointement aux sorties de table d'opacité. ***Cause possible***: oscillations au moment de la mise à l'équilibre du deutérium. ***Remède***: Cesam2k20 "force la convergence" qui, souvent au niveau de quelques 10^{-5}, n'est pas mauvaise.

12. Oscillations violentes de la quantité $A = \frac{1}{\Gamma_1}\frac{\partial \ln P}{\partial \ln R} - \frac{\partial \ln \rho}{\partial \ln R}$ d'un fichier `mon_modele-*.osc` *cf.* §C.1 (Page 345). Ces oscillations ont plusieurs causes décrites, *cf.* §6.9.9 (Page 121). Bien que beaucoup d'efforts aient été consacrés à cette difficulté, jusqu'à présent, le moins mauvais remède consiste à supprimer "à la main" les couches incriminées. La diffusion des éléments chimique évite en grande partie ce phénomène. Lorsque celle-ci n'est pas requise et qu'une bonne représentation de la fréquence de **v** est nécessaire, dans le fichier de données, *cf.* §3.3 (Page 14), coder `'sa'` pour le type de précision, et introduire un peu de diffusion turbulente :

```
/
&NL_DIFF
DIFFUSION=.TRUE.,
NOM_DIFFM='diffm_0',
NOM_DIFFT='difft_nu',
D_TURB=10.d0,
```

```
RE_NU=1.d0,
NOM_FRAD='no_frad'
/
```

qui aura pour effet d'éliminer les disconinuités de la dérivée première de la composition chimique. Le coût à payer est un effort de calcul nettement plus important qu'avec `DIFFUSION=.FALSE.`. La figure 5.1 (Page 41) représente le profil de la fréquence de **v** obtenu pour un modèle de $10M_\odot$ de séquence principale ($X_c = 0.35$), calculé avec la prodédure qui vient d'être décrite. Au cours de l'évolution le cœur convectif a régressé de $0.22R_\star$ à $0.14R_\star$.

13. Mauvaise convergence pour la restitution de l'atmosphère. Ce problème se produit parfois lors de la reprise d'un modèle. La difficulté est due au fait que l'atmosphère qui a été reprise en binaire n'est pas suffisamment proche de celle que Cesam2k20 désire calculer. ***Remède***: supprimer le modèle d'atmosphère en binaire `mon_modele.atm`. Pour initialiser, Cesam2k20 utiliser l'atmosphère standard.

14. Mauvaise convergence, pas temporel tendant vers 0, et/ou oscillations de la vitesse angulaire au voisinage des limites zones radiatives / zones convectives. Les causes ne sont pas identifiées. Une conséquence est la génération d'instabilités de composition chimique, qui se transmettant à la densité, génèrent des coefficients très bruités pour les équations de la diffusion du moment cinétique. Une augmention du coefficient de diffusion turbulente `d_turb`, *cf.* §**??** (Page **??**), permet, parfois, de faire disparaître d'anomalie.

15. Profil très bruité du taux de variation de poids moléculaire moyen Λ, *cf.* §6.8.5 (Page 102), au voisinage des limites zones radiatives / zones convectives. Remède inconnu. Cette anomalie n'a, en général, pas de conséquence directe sur le profil de la vitesse angulaire.

Part II

Description of Cesam2k20

Chapter 6

Physics to numerics

Contents

In this chapter, the physics implemented in Cesam2k20 is presented together with the numerical methods used. The aim is to give as much information as possible, in order to facilitate the understanding of both the physics implemented and the numerical methods used. Some parts are of an elementary level, but any further development of Cesam2k20 is at this price.

The stellar evolution equations are a non-linear integro-differential partial differential problem: it is a boundary problem with initial conditions. Only one spatial dimension is taken into account in the equations.

The method of resolution chosen, the method of "lines" is classical, see Henrici (1962). It consists, for each time step, in solving iteratively and in a decoupled way, the problem of initial conditions (Cauchy problem) then the boundary problem (Dirichlet problem) - a coupled resolution is only possible if the number of chemical species whose temporal evolution must be followed is reduced, which is not the case in general.

The non-linear boundary problem is solved by Newton-Raphson iterations referred to as the "Henyey method" in internal structure (Henyey et al., 1959).

The initial value problem is a steep one, as the various time scales involved differ by several decades.

A numerical difficulty is due to the physical assumption that convective motions homogenise the chemical species and stiffen the rotation. As a result, the continuity of some unknown functions is destroyed at the limits RZ/CZ.

6.1 Integration by splines-collocation

In this section, the numerical integration method used in Cesam2k20 to solve the boundary problems of quasi-static equilibrium and atmospheric restitution is described. It uses techniques and algorithms that are, for the most part, not widely used in stellar evolution. In order to be accessible without too much investment in numerical analysis, but with the risk of plagiarising specialised textbooks, an overview of the thinking that led to the selected algorithms is given before describing them.

This method consists in developing the unknown functions on a basis of piecewise polynomials, the B-splines, and writing that these developments satisfy the boundary conditions and the differential equations at a number *ad hoc* of points. Since the basis functions do not satisfy the boundary conditions, this method is called "quasi-spectral" in the literature.

6.1.1 Standardised B-splines

Given a partition of $[a, b] \subset$: $a = x_0 < x_1 < \ldots < x_n = b$, the set of piecewise polynomials of order m (degree $+1$), which connect in $x_1, x_2, \ldots, x_{n-1}$ is a finite-dimensional vector space M, of which a basis is that of the B-splines. It will be noted in the following: $\{N_i^m\}_{i=1}^M$. For $m = 2$ and $n = 4$ these are the "hat functions" of the Fig. 6.1 (Page 52) and the bell curves for $m = 4$ and $n = 7$.

The various mathematical definitions of these functions are abstruse: i-th divided difference of the translated power function, for example. They can be found in de Boor (1978)p. 108, Schumaker (1981), p. 118, Trenoguine (1980), p. 341, and in many other books. These basic functions, despite their complicated algebra, are very well suited to numerical computation because the computations are easy, stable and efficient. In particular:

- the B-splines are functions with bounded support;

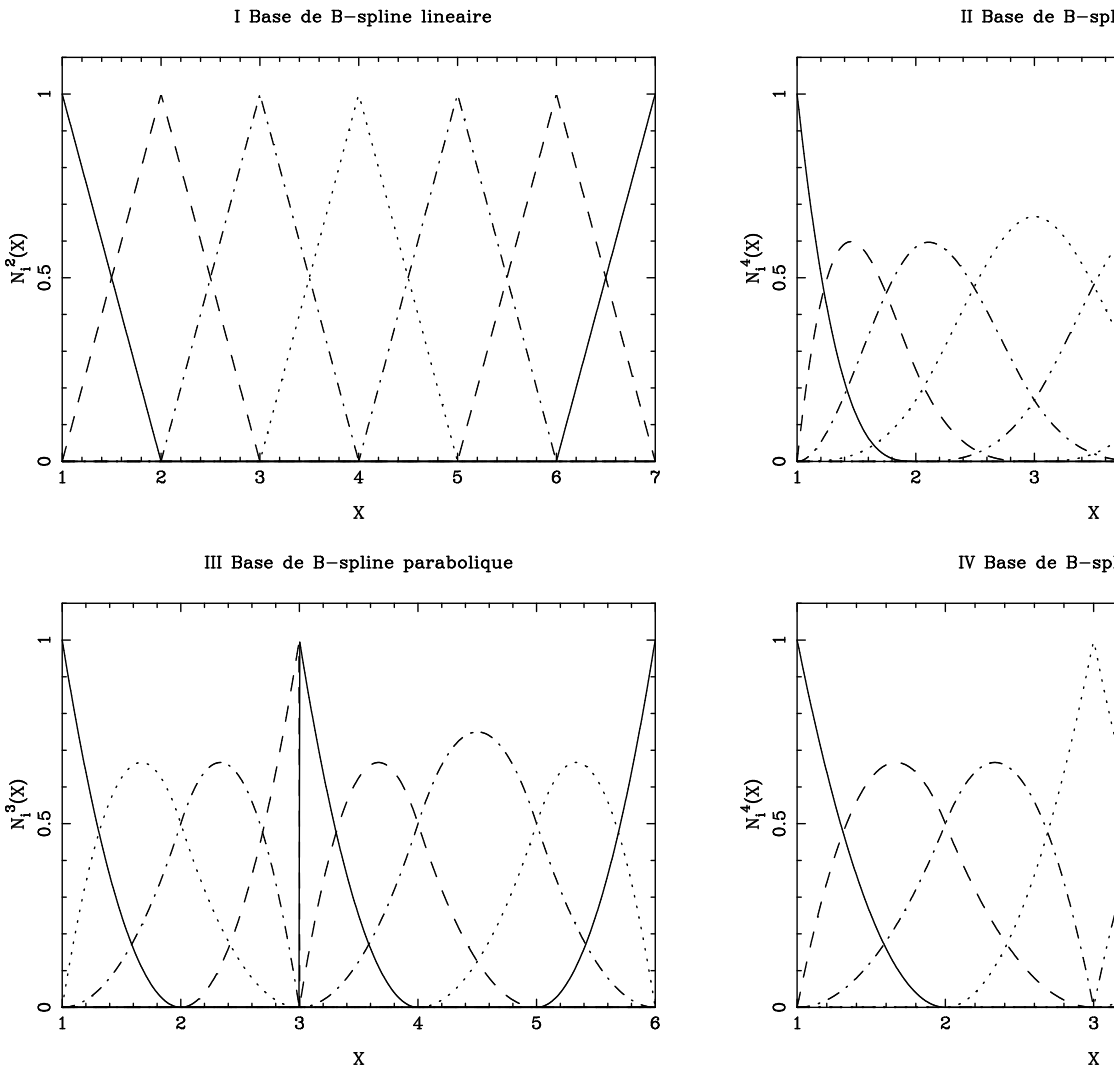


Figure 6.1: Normalized B-splines of order $m = 2$ for $n = 4$ equidistant grid points.

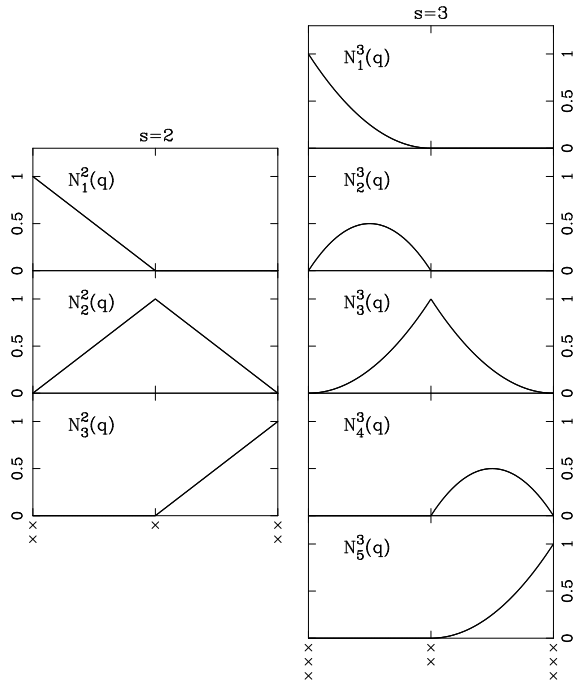


Figure 6.2: Normalized B-splines of order $s = 2$ (left) and $s = 3$ (right) computed with the C. de Boor basis for $n = 3$ equidistant grid points. The nodal vectors are shown as crosses under each graph.

- at a point of their support there are at most m B-splines of order m non-zero, a consequence is the band structure of the linear systems appearing in the calculations;
- a B-spline is always positive or zero;
- the derivative of a B-spline is a linear combination of B-splines of immediately lower order;
- There are efficient and stable numerical algorithms for calculations with B-splines.

6.1.2 Solving a differential problem

The principle of using B-splines to solve a differential problem is that of any spectral method: the unknown functions are developed on a finite basis, in this case that of the B-splines. To calculate the coefficients, we write that these developments satisfy the differential equations and the boundary conditions. To do this there is an alternative:

- The equations are satisfied at an *ad hoc* number of points, the collocation is used to solve the quasi-static equilibrium (see Sect. 6.2) and to restore the atmosphere (see Sect. 6.3.2).
- The basis is imposed to be orthogonal to its image by the differential operator (*i.e.* to the residue), it is the finite element method (Galerkin) which is used to solve the diffusion equations of the chemical species (see Sect. 6.7), and of the angular momentum (see Sect. 6.8.

Thus, for the differential boundary problem:

$$f''(x) = \Phi\left(x; f, f'\right), \qquad \phi_1\left(x_1; f, f'\right) = \alpha, \qquad \phi_2\left(x_n; f, f'\right) = \beta, \qquad \text{sur } [x_1, x_n] \subset \tag{6.1}$$

where ϕ_1, ϕ_2 are functions and α, β are given real numbers. Using the basis $\{N_i^m\}_{i=1}^M$ of B-splines of order $m \geq 4$, on a partition of $[x_1, x_n]$ to be specified, we have:

$$f(x) \simeq \sum_{i=1}^{M} f_i N_i^m(x), \qquad f'(x) \simeq \sum_{i=1}^{M} f_i \frac{\mathrm{d}^2 N_i^m(x)}{\mathrm{d}x^2}, \qquad f''(x) \simeq \sum_{i=1}^{M} f_i \frac{\mathrm{d}^2 N_i^m(x)}{\mathrm{d}x^2}. \tag{6.2}$$

With the collocation method, the M coefficients f_i are obtained by writing that the development of f satisfies the 2 boundary conditions and the equations:

$$\sum_{i=1}^{M} f_i \frac{\mathrm{d}^2 N_i^m(z_c)}{\mathrm{d}x^2} = \Phi(z_c; f, f') \tag{6.3}$$

in $M-2$ points $z_c \in]x_1, x_n[$.

The resulting solution is in $O(||\Delta x||^m)$. If we adopt a judicious choice for the collocation points the solution is in $O(||\Delta x||^{2(m-1)})$: this is the superconvergence (de Boor, 1978, chap. XV).

With Galerkin's method the coefficients f_i would, for example, be obtained by writing that the development of f satisfies the M equations:

$$\sum_{i=1}^{M} f_i \left\langle \frac{\mathrm{d}^2 N_i^m}{\mathrm{d}x^2} \cdot N_l^m \right\rangle = \left\langle \Phi(x_l; f, f') \cdot N_l^m \right\rangle, \quad 1 < l < M \tag{6.4}$$

where $\langle g \cdot h \rangle$ represents a scalar product to be defined. The boundary conditions are introduced by integration by parts (Trenoguine, 1980; Marchouk & Agochkov, 1985).

With both methods, if phi is linear, the coefficients f_i are obtained by solving a linear system. In the case of stellar evolution, it is necessary to use an iterative method *e.g.* fixed point or Newton-Raphson method as the equations are non-linear.

6.1.3 Cesam2k20's architecture

As the algebra of B-splines is quite complex, the architecture of Cesam2k20 is based on a very distinct separation between the physical space and the space of the basis functions. The equations to be solved are written in a form close to their formulation in the physical space. In the B-spline space these equations are solved for the coefficients of the basis functions. Routines serve as interfaces between the two spaces. For example, the coefficients of the quasi-static equations are calculated in physical space by the routine `static_m`. The equations are solved in spline space by the routine `coll_qs`, with the routine `resout` acting as an interface between the two spaces, see Fig. **??**.

6.1.4 Collocation of a non-linear system

The spline/collocation method is explained for the system of n_e ordinary, nonlinear, first-order differential equations with boundary conditions at the ends of the interval of definition *e.g.* the equations of the internal structure. These systems can be written formally:

$$\mathcal{F}_j(x; \mathbf{f}, \mathbf{f}') = 0, \qquad x \in [x_1, x_n], \qquad \mathbf{f} = {}^T(f_1, f_2, \ldots, f_{n_e}) \qquad 1 \leq j \leq n_e \tag{6.5}$$

with boundary conditions:

$$\alpha_q\left(\mathbf{f}(x_1)\right) = 0, \forall q \in Q, \; Q \subseteq J = \{1, 2, \ldots, n_e\}, \qquad \beta_r\left(\mathbf{f}(x_n)\right) = 0, \; \forall r \in \boldsymbol{\mathcal{C}}_J Q \tag{6.6}$$

the notation $\mathcal{C}_J Q$ meaning the complementary of Q in the index set J. By replacing each component f_j of $\mathbf{f}$, $1 \leq j \leq n_e$, by its development on the basis of B-splines:

$$f_j(x) = \sum_{i=1}^{M} f_{i,j} N_i^m(x), \qquad f_j'(x) = \sum_{i=1}^{M} f_{i,j} \frac{\mathrm{d}N_i^m(x)}{\mathrm{d}x} \tag{6.7}$$

the unknowns are now the real $f_{i,j}$. They verify:

$$\begin{cases} \mathcal{F}_j \left(x; \sum_{i=1}^{M} f_{i,1} N_i^m(x), \ldots, \sum_{i=1}^{M} f_{i,n_e} N_i^m(x), \sum_{i=1}^{M} f_{i,1} \frac{\mathrm{d}N_i^m(x)}{\mathrm{d}x}, \ldots, \sum_{i=1}^{M} f_{i,n_e} \frac{\mathrm{d}N_i^m(x)}{\mathrm{d}x} \right) = 0, \forall x \in]x_1, x_n[, \forall j \in J \\ \alpha_q \left(\sum_{i=1}^{M} f_{i,1} N_i^m(x_1), \ldots \sum_{i=1}^{M} f_{i,n_e} N_i^m(x_1) \right) = 0, \forall q \in Q \\ \beta_r \left(\sum_{i=1}^{M} f_{i,1} N_i^m(x_n), \ldots \sum_{i=1}^{M} f_{i,n_e} N_i^m(x_n) \right) = 0, \forall r \in \mathcal{C}_J Q \end{cases} \tag{6.8}$$

For a linear differential system, the $f_{i,j}$ come together. They are evaluated by writing Eq. (6.7) at $M-1$ collocation points $z_c \in]x_1, x_n[$ and solving the resulting linear system. The equations of the internal structure are not linear. The Newton-Raphson iterative method is used to solve the system Eq. (6.27). which requires an initial solution $\mathbf{f}^0$. Noting $\mathbf{f}^p$ the solution obtained at iteration $p \geq 0$, the linearized equations are:

$$\begin{cases} \mathcal{F}_j \left(z_c; \sum_{i=1}^{M} f_{i,1}^p N_i^m(z_c), \ldots, \sum_{i=1}^{M} f_{i,n_e}^p N_i^m(z_c), \sum_{i=1}^{M} f_{i,1}^p \frac{\mathrm{d}N_i^m(z_c)}{\mathrm{d}x}, \ldots, \sum_{i=1}^{M} f_{i,n_e}^p \frac{\mathrm{d}N_i^m(z_c)}{\mathrm{d}x} \right) \\ = \sum_{k=1}^{n_e} \sum_{i=1}^{M} \left(\frac{\partial \mathcal{F}_j}{\partial f_k} N_i^m(z_c) + \frac{\partial \mathcal{F}_j}{\partial f_k'} \frac{\mathrm{d}N_i^m(z_c)}{\mathrm{d}x} \right) df_{i,k}^p, \ \forall j \in J, \ 1 \leq c < M \\ \alpha_q \left(\sum_{i=1}^{M} f_{i,1}^p N_i^m(x_1), \ldots \sum_{i=1}^{M} f_{i,n_e}^p N_i^m(x_1) \right) = \sum_{k=1}^{n_e} \frac{\partial \alpha_q}{\partial f_k} \sum_{i=1}^{M} N_i^m(x_1) df_{i,k}^p, \forall q \in Q, \\ \beta_r \left(\sum_{i=1}^{M} f_{i,1}^p N_i^m(x_n), \ldots \sum_{i=1}^{M} f_{i,n_e}^p N_i^m(x_n) \right) = \sum_{k=1}^{n_e} \frac{\partial \beta_r}{\partial f_k} \sum_{i=1}^{M} N_i^m(x_n) df_{i,k}^p, \forall r \in \mathcal{C}_J Q \end{cases} \tag{6.9}$$

The sums on i are, in fact, only on m indices since at a point x there are, at most, m non-zero B-splines. The system (6.9) is linear for the corrections $df_{i,k}^p$. We will have:

$$f_{i,k}^{p+1} = f_{i,k}^p - df_{i,k}^p, 1 \leq i \leq M, 1 \leq k \leq n_e \tag{6.10}$$

Without diffusion, due to convective mixing, at the RZ/CZ boundaries the chemical composition shows a discontinuity, resulting in a density discontinuity. With diffusion, the discontinuities affect only the first derivatives. These discontinuities led to the choice of pressure as the dependent variable instead of density. Moreover, the use of the latter would require the evaluation of the partial derivatives $\partial P/\partial Rho|_{T,X}$, $\partial P/\partial T|_{Rho,X}$ and $\partial P/\partial X|_{T,\rho}$ in the first two equations of the system ((**??**)), and not just the "thermodynamic" second derivatives:

$$\left.\frac{\partial^2 P}{\partial T^2}\right|_{\rho,X}, \qquad \left.\frac{\partial^2 P}{\partial \rho^2}\right|_{T,X}, \qquad \left.\frac{\partial^2 P}{\partial \rho \partial T}\right|_{X}, \qquad \left.\frac{\partial^2 P}{\partial \rho \partial X}\right|_{\mathrm{T}}, \qquad \left.\frac{\partial^2 P}{\partial T \partial X}\right|_{\rho} \tag{6.11}$$

The quasi-static equations are integrated with a scheme assuming continuity of each dependent variable with, possibly, a discontinuity of the first derivative, see Sect. 6.1.5. Without diffusion, due to its discontinuous nature, in the vicinity of the RZ/CZ boundaries, the profile of γ will show oscillations, but these are not important because, without diffusion, γ is not coupled to the other variables. It is not the same with diffusion, but then γ is continuous, non-derivable at the RZ/CZ limits, a situation taken into account by the integration scheme.

6.1.5 Choice of basis for collocation

As they are defined in de Boor (1978, p. 280), a particular choice of the base and of the collocation points makes it possible to obtain an order of precision higher than that of the interpolation functions, it is the *superconvergence*. We give ourselves the order of the m B-splines and the grid of the connection points. The pieces of polynomials are connected at these points by ensuring the continuity of the B-spline and its derivatives up to an order one unit lower than that of the differential system. In Cesam2k20 we use the equidistant grid $\{1, 2, \ldots n\}$, the order of the differential equations is $r = 1$ and the order of the B-splines is fixed by the type of precision indicated in the data file (see Sect. 3.3), or, possibly in a file `reglages` (see Sect. 3.16).

Between two grid points, the dependent variables are interpolated by pieces of polynomials of order $m + r = 2$ (straight lines), or $m + r = 3$ (parabolas). The pieces of polynomials are connected to the grid points by continuity, their first derivatives being able to be, if necessary, different. This particularity is exploited to describe correctly the profile of the density at the RZ/CZ limits (see Sect. 6.2.1.

Figure 3 shows this basis for $n = 3$. At n points on the grid, the solution is of order $2m = 2$ or 4; this superconvergence is due to this particular arrangement of collocation points. Between the connection points the solution is of order $m + r = 2$ or 3; the superconvergence cannot therefore be exploited directly at any point. If it is necessary to know the solution at points other than those defined by the grid, in order to exploit the superconvergence, it is necessary to make an interpolation of order $2m$ [1].

Under these conditions, the dimension of the basis of B-splines is $M = (n-1)m + r$. This is the number of coefficients $f_{i,j}$ to be calculated for each unknown (index j). An equal number of condition equations must be collected and there are as many boundary conditions as there are unknowns. We therefore need $(n-1)m$ collocation points, i.e. m collocation points between two connection points. These points are located at the abscissae of the zeros of the Legendre polynomial of degree m, defined on each interval reduced to $[-1, 1]$; that is ± 0.57735 and 0 for $m = 2$. The superconvergence results from this arrangement. Once these points have been fixed, we save computational time by determining, once and for all, the values of the B-splines and their first derivatives.

Cesam2k20 uses the algorithm of de Boor (1978), according to the version of Schumaker (1981, chap. 5), to compute the value of all non-identically zero B-splines at any point. This algorithm is based on a split-difference calculation for a Hermite interpolation, which is the object of the routines `bval0, bval1, bvald`, (see Sect. 8.2). Its practical arrangement is based on the construction of a table of abscissas where some are repeated several times. It constitutes the *nodal vector* constructed in the routine `coll` (see Sect. 8.2.9).

At each collocation point, the order of interpolation being $m + r = 3$ or $m + r = 2$, there are 3 or 2 non-zero B-splines. Each of the Eq. (6.27) involves 3 or 2 non-identical zero coefficients per unknown and, depending on whether $m = 1$ or $m = 2$ is used, there are 1 or 2 collocation points between two connection points, the structure of the Jacobian of the Newton-Raphson method is as follows, since the basis functions have a bounded support

[1] The experience of the calculations showed that the exploitation of the superconvergence was delicate because of the discontinuities due to the displacements of the limits RZ/CZ.

$$m = 1 : \quad \begin{pmatrix} \times & \times & & & & & \\ \circ & \circ & & & & & \\ & \times & \times & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & \times & \times & \\ & & & & & \times & \times \\ & & & & & \circ & \circ \end{pmatrix} \tag{6.12}$$

$$m = 2 : \quad \begin{pmatrix} \times & \times & \times & & & & & & & \\ \times & \times & \times & & & & & & & \\ \circ & \circ & \circ & & & & & & & \\ & & & \times & \times & \times & & & & \\ & & & \times & \times & \times & & & & \\ & & & & & \cdot & & & & \\ & & & & & & \cdot & & & \\ & & & & & & & \cdot & & \\ & & & & & & & \times & \times & \times \\ & & & & & & & \times & \times & \times \\ & & & & & & & & & \cdot \\ & & & & & & & \times & \times & \times \\ & & & & & & & \times & \times & \times \\ & & & & & & & \circ & \circ & \circ \end{pmatrix} \tag{6.13}$$

Each collocation point corresponds to a line of $\times$. Each of them[2], represents a square matrix whose order is the number of unknowns $n_e = 7$, $n_e = 8$ with turbulent pressure. For example, with $m = 2$, each interval between two connection points corresponds to a block of $m = 2$ lines of $(m + r)\times = (2 + 1)\times = 3\times$. Two consecutive blocks are connected by $r = 1$ column of blocks. The lines corresponding to the boundary conditions are represented by $\circ$. These are matrices of format $3 \times n_e$, since there are three boundary conditions at each end of the integration interval $[1, n]$.

For the internal structure, the linear system is formed and solved in the subroutine `coll_qs` (see Sect. 7.33.1) from the coefficients calculated in `static_m` (see Sect. 7.76), or `static_r` (see Sect. **??**). For the restitution of the atmosphere, these are the routines `coll_atm` (see Sect. 7.9.2) and `eq_atm` (see Sect. 7.10.1) respectively.

We can notice that it is enough to keep in memory lines of length $(m + r)n_e = (2 + 1)6 = 18$ to solve the system. It has $n_e((n - 1)m + r) = 6((150 - 1)2 + 1) = 1794$ lines for $n = 150$. This remark is exploited in the routine `gausdp_g` used to solve the system by Gaussian elimination with partial pivot (see Sect. 8.1).

If turbulent pressure is not taken into account an equation is omitted.

6.2 Internal structure equations

Cesam2k20 solves the internal structure equations in a form that is quite different from their classical expression. In this paragraph, we establish the relations used, then we give the change of variable allowing to adapt automatically the network to the evolution of the variations of the unknowns, finally we describe the method used to fix a point of the discretization grid at each limit between a radiative zone and a convective zone.

[2]If we do not take into account the turbulent pressure

With, as an independent Lagrangian variable, the mass M contained inside a sphere of radius R, the stellar evolution equations are (e.g. Kippenhahn & Weigert, 1991, Sect. 9.1):

$$\left\{\begin{array}{rcl} \frac{\partial P}{\partial M} & = & -\frac{\mathcal{G}M}{4\pi R^4}+\frac{\Omega^2}{6\pi R}-\frac{1}{4\pi R^2}\frac{\mathrm{D}V}{\mathrm{D}t} \\ V & = & \frac{DR}{Dt} \\ \frac{\partial T}{\partial M} & = & \frac{\partial P}{\partial M}\frac{T}{P}\nabla \\ \frac{\partial R}{\partial M} & = & \frac{1}{4\pi R^2\rho} \\ \frac{\partial L}{\partial M} & = & \epsilon-\epsilon_{\mathrm{G}}=\epsilon-\frac{\mathrm{D}U}{\mathrm{D}t}+\frac{P}{\rho^2}\frac{\mathrm{D}\rho}{\mathrm{D}t} \\ P & = & P_{\mathrm{gas}}+P_{\mathrm{tur}} \\ \frac{\partial X_{\mathrm{i}}}{\partial t} & = & -\frac{\partial F_i}{\partial M}+\Psi_i(P_{\mathrm{gas}},T;\mathcal{X}),1\leq i\leq n_{\mathrm{elem}}. \end{array}\right. \tag{6.14}$$

Boundary conditions:

$$\begin{array}{rcl} R(0,t) & = & 0, \\ L(0,t) & = & 0, \\ P(M_{\mathrm{b}},t) & = & P_{\mathrm{b}}(L,R), \\ P_{\mathrm{gas}}(M_{\mathrm{b}},t) & = & P_{\mathrm{gaz,b}}(L,R), \\ T(M_{\mathrm{b}},t) & = & T_{\mathrm{b}}(L,R). \end{array}$$

Initial conditions (for $0\leq M\leq M_{\mathrm{b}}$):

$$\begin{array}{rcl} X_i(M,0) & = & X_{i,0}, i=1,\ldots,n_{\mathrm{elem}}, \\ P(M,0) & = & P_0(M), \\ P_{\mathrm{gas}}(M,0) & = & P_{\mathrm{gaz},0}(M), \\ T(M,0) & = & T_0(M), \\ R(M,0) & = & R_0(M), \\ L(M,0) & = & L_0(M). \end{array}$$

The classical notations were used. R is the radius (distance to the centre of the star), $P\equiv P_{\mathrm{tot}}$ the total pressure, P_{gas} the gas pressure, P_{tur} the turbulent pressure, ρ the density, Ω the angular velocity, $\mathcal{G}$ the gravitational constant, T the temperature, $\nabla\equiv\partial\ln T/\partial\ln P|_t$ the gradient, L the luminosity, $\epsilon=\epsilon_{\mathrm{nu}}-\partial\epsilon_\Omega/\partial t-\epsilon_\nu$ the energy flow rate resulting from nuclear reactions, local momentum dissipation (see Sect. 6.6), and neutrino losses, ϵ_{G} is the gravitational energy, (6.62), M the mass included in the sphere of radius R, t the time, X_i the abundance of the chemical element i, F_i the scattering flux for chemical element i, $\mathcal{X}=\{X_i\}$ the chemical composition vector, Ψ_i the rate of change of abundance of chemical element i by thermonuclear reactions, n_{elem} the number of chemical species considered. $M_{\mathrm{b}}(R,L)$, $P_{\mathrm{b}}(L,R)$, $P_{\mathrm{gaz,b}}(L,R)$ and $T_{\mathrm{b}}(L,R)$ are four functions describing respectively the variations of the mass, the total pressure, the gaseous pressure and the temperature, as a function of the radius and the luminosity at the external limit of the envelope which coincides with the base of the atmosphere. The 5 functions: $P_0(M)$, $P_{\mathrm{gaz},0}(M)$, $T_0(M)$, $R_0(M)$ and $L_0(M)$ describe, as a function of mass, respectively the total pressure, the gas pressure, the temperature, the radius and the luminosity of the zero-age model.

This formulation assumes spherical symmetry and does not take into account the magnetic field, and rotation is only involved through effective gravity. The equations describing the diffusion of chemical elements and angular momentum will be described in Sect. 6.7 and Sect. 6.8 respectively.

6.2.1 Density discontinuities

The time scale of the reversal of convective elements being small compared to the nuclear time scale, it is assumed that convection homogenises the CZs (see Sect. 6.8.7. The chemical composition, and thus the density, are *discontinuous* at the boundaries RZ/CZ[3]. Cesam2k20 uses nodal vectors for the various integrations (structure, chemical composition, angular momentum) constructed in such a way that the points where the equations are formed do not coincide with the points of discontinuity, so that the latter are implicitly taken into account.

6.2.2 Integration variables

The natural variables with respect to which the Eqs (6.14) are written are not well suited to numerical calculation. In particular, the derivatives of pressure and radius have singularities at $M \equiv 0$, i.e. at the centre.

These singularities disappear by using the radius R as an independent variable. The external boundary condition then becomes a free boundary condition, since the total radius is not fixed. Although this free limit is not a difficulty for the numerical method used in Cesam2k20, the radius is not, however, the best choice for the independent variable; indeed, it can be shown (Morel, 1997) that the singularities are overcome and that an optimal use of the numerical approximation is obtained when the following variables are used for mass, radius and luminosity respectively: $M^{2/3}$, R^2 and $L^{2/3}$. It has been found convenient to use variables normalized to solar values:

$$\xi = \ln P,\ \xi_g = \ln P_{\rm gas},\ \eta = \ln T, \mu = \left(\frac{M}{M_\odot}\right)^{\frac{2}{3}}, \zeta = \left(\frac{R}{R_\odot}\right)^2, \lambda = \left(\frac{L}{L_\odot}\right)^{\frac{2}{3}}, \gamma = \ln \rho \tag{6.15}$$

With these new variables, the system (6.14)) becomes:

$$\begin{aligned}
\frac{\partial \xi}{\partial \mu} &= \left[-\frac{3G}{8\pi}\left(\frac{M_\odot}{R_\odot^2}\right)^2\left(\frac{\mu}{\zeta}\right)^2 + \frac{M_\odot}{4\pi R_\odot}\sqrt{\frac{\mu}{\zeta}}\Omega^2 - \frac{3M_\odot}{8\pi R_\odot^2}\frac{\sqrt{\mu}}{\zeta}\frac{\mathrm{D}V}{\mathrm{D}t}\right]\exp(-\xi)\\
V &= \frac{R_\odot}{2\sqrt{\zeta}}\frac{\mathrm{D}\zeta}{\mathrm{D}t}\\
\frac{\partial \eta}{\partial \mu} &= \frac{\partial \xi_t}{\partial \mu}\nabla\\
\frac{\partial \zeta}{\partial \mu} &= \frac{3}{4\pi}\frac{M_\odot}{R_\odot^3}\frac{1}{\rho}\sqrt{\frac{\mu}{\zeta}}\\
\exp(\xi) &= \exp(\xi_g) + P_{\rm tur}\\
\gamma &= \ln\rho(\exp(\xi_g), \exp(\eta), X_i)\\
\frac{\partial \lambda}{\partial \mu} &= \frac{M_\odot}{L_\odot}\sqrt{\frac{\mu}{\lambda}}\left(\epsilon - \frac{\mathrm{D}U}{\mathrm{D}t} + \frac{P}{\rho^2}\frac{\mathrm{D}\rho}{\mathrm{D}t}\right)\\
\frac{\partial X_i}{\partial t} &= -\frac{2}{3M_\odot\sqrt{\mu}}\frac{\partial F_i}{\partial \mu} + \Psi_i(\xi_t, \eta; \mathcal{X}),\ 1 \le i \le n_{\rm elem}
\end{aligned}$$

[3]When diffusion of chemical elements is taken into account, the chemical composition and density are continuous and non-derivative at these boundaries.

Boundary conditions:

$$\begin{aligned}
\zeta(1,t) &= 1,\\
\lambda(1,t) &= 1,\\
\xi(\mu_{\rm ext},t) &= \ln\left[P_{\rm ext}\left(\sqrt{\lambda}^3 L_\odot, \sqrt{\zeta} R_\odot\right)\right]\\
\xi_{\rm g}(\mu_{\rm ext},t) &= \xi(\mu_{\rm ext},t),\\
\eta(\mu_{\rm ext},t) &= \ln\left[T_{\rm ext}\left(\sqrt{\lambda}^3 L_\odot, \sqrt{\zeta} R_\odot\right)\right]\\
\gamma &= \ln\rho(P_{\rm ext}, T_{\rm ext}, X_i).
\end{aligned}$$

Initial conditions (for $1 \leq \mu \leq \mu_{\rm ext}$):

$$\begin{aligned}
X_i(\mu,0) &= X_{i,0},\ i = 1,\ldots,n_{\rm elem}\\
\xi(\mu,0) &= \xi_0(\mu),\\
\xi_{\rm g}(\mu,0) &= \xi_{{\rm g}\,0}(\mu),\\
\eta(\mu,0) &= \eta_0(\mu),\\
\gamma(\mu,0) &= \gamma_0(\mu),\\
\zeta(\mu,0) &= \zeta_0(\mu),\\
\lambda(\mu,0) &= \lambda_0(\mu).
\end{aligned}$$

At the centre, the singularity of the gradients is lifted:

$$\lim_{\mu\to 0}\left\langle\frac{\partial\xi}{\partial\mu}\right\rangle = \frac{\mathcal{G}}{2}\left(\frac{4\pi M_\odot^2}{3}\right)^{1/3}\exp\left(-\xi(0,t)\right)\rho(0,t)^{4/3} < \infty, \tag{6.16}$$

$$\lim_{\mu\to 0}\left\langle\frac{\partial\zeta}{\partial\mu}\right\rangle = \left(\frac{3M_\odot}{4\pi R_\odot^3}\right)^{2/3}\rho^{-2/3}(0,t) < \infty, \tag{6.17}$$

and, moreover:

$$\lim_{\mu\to 0}\left\langle\frac{\mu}{\lambda}\right\rangle = \lim_{\mu\to 0}\left\langle\frac{\partial\lambda}{\partial\mu}\right\rangle = \frac{M_\odot}{L_\odot}\left[\epsilon - T\left(\frac{\partial S}{\partial t}\right)_\mu\right], \tag{6.18}$$

$$\lim_{\mu\to 0}\frac{\mu}{\zeta} = \left(\frac{4\pi}{3}\right)^{2/3}\left(\frac{R_\odot^2}{M_\odot}\right)^{2/3}\rho^{2/3}, \tag{6.19}$$

quantities that can be derived at least once. Without the $2/3$ exponent affecting the luminosity, the second member of the energy equation would have been:

$$\frac{\partial\lambda}{\partial\mu} = \frac{M_\odot}{L_\odot}\mu^{1/2}\left[\epsilon - \frac{{\rm D}U}{{\rm D}t} + \frac{P}{\rho^2}\frac{{\rm D}\rho}{{\rm D}t}\right] \tag{6.20}$$

the square root, poorly represented by a polynomial, in the vicinity of $\mu = 1$, leading to numerical instability. The exponent 2/3 thus allows a better numerical representation of the luminosity in the vicinity of the centre, but prevents this dependent variable from becoming negative, which, in certain phases of the evolution, can become prohibitive. When this eventuality can occur, although less satisfactory numerically, it is necessary to use the radius as an independent Eulerian variable and, in this case, the set of variables used by Cesam2k20 is:

$$\xi = \ln P, \xi_{\rm g} = \ln P_{\rm gas}, \eta = \ln T, \mu = \left(\frac{M}{M_\odot}\right), \zeta = \left(\frac{R}{R_\odot}\right), \lambda = \left(\frac{L}{L_\odot}\right), \gamma = \ln\rho. \tag{6.21}$$

6.2.3 Adaptation of discretization

The areas in which the solutions vary rapidly move during the evolution, so that the initial network becomes more or less quickly inadequate. In order not to degrade the accuracy, it is then necessary to tighten the nodes in the regions with strong variations, and loosen them on the plateaus. There are two solutions to this difficult numerical problem:

- one or more layers are added/removed when the increment of a variable is too/not enough (Kippenhahn et al., 1967).
- the nodes are arranged automatically so as to satisfy certain criteria (Eggleton, 1971; Press et al., 1995).

Cesam2k20 uses the second solution. The idea consists in carrying out a change of variable. We use as dependent variable a function, to be defined, of the dependent variables whose variations we wish to control, a function to which we impose a constant variation from one node of the grid to the next. Initially, the total number of nodes is fixed. These are arranged automatically by the calculation at the correct abscissae. The mass, an independent variable, becomes a dependent variable whose value is known only at the end of the integration. To this end, the spacing function $Q(\mu, t)$ is defined on the basis of the knowledge that we have, a priori, of the solution. For example, for the internal structure, we could take[4]: $Q(\mu, t) = \ln P$. At time t we are looking for a distribution of abscissas $\mu_i, i = 1, \ldots, n$ such that the increment of the spacing function is constant from one node to the other of the grid. One will thus seek to achieve:

$$Q(\mu_i, t) - Q(\mu_{i+1}, t) = \mathrm{Cte}(t), i = 1, \ldots, n-1 \tag{6.22}$$

This imposes a strict monotonicity on the spacing function $Q(\mu, t)$. The number of layers n being given, for t fixed, one introduces the "index" function $q(\mu, t)$ applying the interval of variation of $\mu : [0, \mu_{\mathrm{ext}}] \subset \mathbb{R}$ on $[1, n]$. The condition (6.22) is then written in an equivalent way:

$$\left.\frac{\partial Q}{\partial q}\right|_t = \mathrm{Cte}(t) = \psi(t) \Longleftrightarrow \left.\frac{\partial^2 Q}{\partial q^2}\right|_t = \left.\frac{\partial \psi}{\partial q}\right|_t = 0 \tag{6.23}$$

and, for the boundary conditions: $\mu = 0$ at $q = n$ and $\mu = \mu_{\mathrm{ext}}$ at $q = n$, since the layer of index 1 corresponds to the centre, and that of index n to the surface.

For t fixed, the change of variable: $\mu \to q(\mu, t)$ allows to write:

$$\left.\frac{\partial Q}{\partial q}\right|_t = \left.\frac{\partial Q}{\partial \mu}\right|_t \left.\frac{\partial \mu}{\partial a}\right|_t = \theta \left.\frac{\partial \mu}{\partial q}\right|_t \tag{6.24}$$

where $\theta(\mu, t) = \partial Q / \partial \mu|_t$ is calculated from the expression of the spacing function $Q(\mu, t)$. There are therefore two additional unknown functions: $\psi(t) = \left.\frac{\partial Q}{\partial q}\right|_t$ and $\mu(q, t)$, solutions of the first order differential system with boundary conditions:

$$\left.\frac{\partial \mu}{\partial q}\right|_t = \frac{\psi}{\theta} \text{ et } \left.\frac{\partial \psi}{\partial q}\right|_t = 0, \text{ with : } \begin{cases} q = 1, \ \mu = 0 \\ q = n, \ \mu = \mu_{\mathrm{ext}} \end{cases} \tag{6.25}$$

These equations are solved simultaneously with the system of Eqs. (6.14) written with respect to the variables q and t; for this purpose, the following relations are available, where $f(\mu, t)$ denotes any dependent variable:

$$\left.\frac{\partial f}{\partial t}\right|_\mu = \left.\frac{\partial f}{\partial t}\right|_q - \left.\frac{\partial f}{\partial \mu}\right|_t \left.\frac{\partial \mu}{\partial t}\right|_q \text{ et } \left.\frac{\partial f}{\partial \mu}\right|_t = \left.\frac{\partial f}{\partial q}\right|_t \left.\frac{\partial \mu}{\partial q}\right|_t^{-1}. \tag{6.26}$$

The abscissas used are $1, 2, \ldots, n$, so we have an *evenly spaced grid*.

[4] This is equivalent to taking the pressure as an independent variable.

6.2.4 Changing the total number of layers

It has been assumed up to now that the total number of layers was fixed once and for all, which is obviously a constraint, since for a given phase of the evolution, there is the risk of working with a too fine discretisation (resp. coarse) with the consequence of useless calculations (resp. imprecise). With an increase (resp. decrease) in the number of layers, the jump [5] in the repartition function from one layer to the next decreases (resp. increases), so that from one time step to the next, it is possible to adjust the number of layers in order to keep $i(t)$ within a fixed range. This arrangement corresponds exactly to the physical meaning that can be attributed to the repartition function, which is to maintain the variations of certain physical quantities within a given interval.

Then $\psi(t) = \partial Q/\partial q|_\mu$ and $\theta(\mu, t) = \partial Q/\partial \mu|_t \Rightarrow \psi/\theta = \partial \mu/\partial q|_t$, and Eqs. (6.14) become:

$$
\begin{aligned}
\frac{\partial \xi}{\partial q} &= \left[-\frac{3\mathcal{G}}{8\pi} \left(\frac{M_\odot}{R_\odot^2} \right)^2 \left(\frac{\mu}{\zeta} \right)^2 + \frac{M_\odot}{4\pi R_\odot} \sqrt{\frac{\mu}{\zeta}} \Omega^2 - \frac{3M_\odot}{8\pi R_\odot^2} \frac{\sqrt{\mu}}{\zeta} \frac{\mathrm{D}V}{\mathrm{D}t} \right] \exp(-\xi) \frac{\psi}{\theta} \\
V &= \frac{R_\odot}{2\sqrt{\zeta}} \frac{\mathrm{D}\zeta}{\mathrm{D}t} \\
\frac{\partial \eta}{\partial q} &= \frac{\partial \xi}{\partial q} \nabla \\
\frac{\partial \zeta}{\partial q} &= \frac{3}{4\pi} \frac{M_\odot}{R_\odot^3} \frac{1}{\rho} \left(\frac{\mu}{\zeta} \right)^{1/2} \frac{\psi}{\theta} \\
\exp(\xi) &= \exp(\xi_g) + P_{\mathrm{tur}} \\
\gamma &= \ln \rho(\exp(\xi_g), \exp(\eta), X_i) \\
\frac{\partial \lambda}{\partial q} &= \frac{M_\odot}{L_\odot} \left(\frac{\mu}{\lambda} \right)^{1/2} \left[\epsilon - \frac{\mathrm{D}U}{\mathrm{D}t} + \frac{P}{\rho^2} \frac{\mathrm{D}\rho}{\mathrm{D}t} \right] \frac{\psi}{\theta} \\
\frac{\partial \mu}{\partial q} &= \frac{\psi}{\theta} \\
\frac{\partial \psi}{\partial q} &= 0 \\
\frac{\partial X_i}{\partial t} &= -\frac{2}{3M_\odot \sqrt{\mu}} \frac{\partial F_i}{\partial \mu} + \Psi_i(\xi_t, \eta; \mathcal{X}), \ 1 \leq i \leq n_{\mathrm{elem}}
\end{aligned}
$$

[5] The $\psi(t)$ is either positive or negative, here it is assumed to be positive.

Boundary conditions:

$$\begin{aligned}
\mu(n,t) &= \mu_{\rm ext}\left(\sqrt{\lambda}^3(n,t)L_\odot, \sqrt{\zeta(n,t)}R_\odot\right),\\
\xi(n,t) &= \ln\left(P_{\rm ext}\left(\sqrt{\lambda}^3(n,t)L_\odot, \sqrt{\zeta(n,t)}R_\odot\right)\right),\\
\xi_{\rm g}(n,t) &= \xi(n,t),\\
\eta(n,t) &= \ln\left(T_{\rm ext}\left(\sqrt{\lambda}^3(n,t)L_\odot, \sqrt{\zeta(n,t)}R_\odot\right)\right),\\
\gamma(n,t) &= \ln\rho(P_{\rm ext}, T_{\rm ext}, X_i)\\
\zeta(1,t) &= 0,\\
\lambda(1,t) &= 0,\\
\mu(1,t) &= 0.
\end{aligned}$$

Initial conditions (for $1 \leq q \leq n$):

$$\begin{aligned}
X_i(q,0) &= X_{i,0}, i = 1,\ldots,n_{\rm elem}\\
\xi(q,0) &= \xi_0(q),\\
\xi_{\rm g}(q,0) &= \xi_{\rm g\ 0}(q),\\
\eta(q,0) &= \eta_0(q),\\
\zeta(q,0) &= \zeta_0(q),\\
\lambda(q,0) &= \lambda_0(q),\\
\gamma(q,0) &= \gamma_0(q).
\end{aligned}$$

These physical space equations are formed in the routine `static_m` (see Sect. 7.76). The spacing function Q must be strictly monotonic; it must therefore be composed only of functions varying in the same direction. It goes without saying that its expression must be as simple as possible, and there is no need to constrain it with "secondary variables" such as $P_{\rm gas}$ or Ω. Since ξ and η vary in the opposite direction to ζ, λ and μ, when there is no reason to favour one variable over another, formally it would be appropriate to use:

$$Q(\mu,t) = \frac{\xi}{\Delta\xi} + \frac{\eta}{\Delta\eta} - \frac{\zeta}{\Delta\zeta} - \frac{\lambda}{\Delta\lambda} - \frac{\mu}{\Delta\mu} \tag{6.27}$$

$$\Delta\xi = \xi(n) - \xi(1), \qquad \Delta\eta = \eta(n) - \eta(1), \qquad \Delta\zeta = \zeta(n) - \zeta(1), \tag{6.28}$$

$$\Delta\lambda = \lambda(n) - \lambda(1), \qquad \Delta\mu = \mu(n) - \mu(1). \tag{6.29}$$

$$\tag{6.30}$$

So we would have:

$$\theta(\mu,t) = \left.\frac{\partial Q}{\partial\mu}\right|_t = \frac{1}{\Delta\xi}\frac{3\mathcal{G}}{8\pi}\left(\frac{M_\odot}{R_\odot^2}\right)^2 \exp(-\xi)\left(\frac{\mu}{\zeta}\right)^2\left(1 + \nabla\frac{\Delta\xi}{\Delta\eta}\right) - \frac{1}{\Delta\zeta}\frac{3}{4\pi}\frac{M_\odot}{R_\odot^3}\frac{1}{\rho}\left(\frac{\mu}{\zeta}\right)^{1/2} - \frac{1}{\Delta\lambda}\frac{M_\odot}{L_\odot}\left(\frac{\mu}{\lambda}\right)^{1/2}\epsilon - 1. \tag{6.31}$$

and similar relationships when the radius is used as an independent variable. The distribution factors $\Delta\xi$, $\Delta\eta$, $\Delta\zeta$, $\Delta\lambda$ and $\Delta\mu$ can be either fixed once and for all or adjusted at the end of each time step so as to follow the evolution.

While this "complete" form of the repartition function was used for versions 0, 1 and 2 of CESAM, it subsequently turned out that $Q \equiv a\xi + b\mu$ (resp. $Q \equiv a\xi + b\zeta$) with Lagrangian variables (resp. eulerian) seemed to be the simplest, most robust and, indeed, most efficient form. The values $a = -1$ and $b \sim 15$ of the distribution factors, defined in the routineindexrepartition function `cesam` (see. 7.7.3), according to the type of precision required, give satisfaction.

The initial approximation of $\psi(t_0) = \partial Q/\partial q|_t$ is obtained by numerically deriving $Q(\mu, t)$ (obtained from the initial provisional solution) with respect to the new independent variable: the discretisation function $q(\mu, t)$.

With the Eulerian independent variable we have: $\psi(t) = \partial Q/\partial q|_\zeta$ and $\theta(\zeta, t) = \partial Q/\partial \zeta|_t$ and the differential problem is written:

$$
\begin{aligned}
\frac{\partial \xi}{\partial q} &= \left[-\frac{GM_\odot}{R_\odot}\frac{\mu}{\zeta^2} + \frac{2R_\odot}{3}\zeta\Omega^2 - \frac{\mathrm{D}V}{\mathrm{D}t}\right]\rho\exp(-\xi)\frac{\psi}{\theta}\\
V &= R_\odot\frac{\mathrm{D}\zeta}{\mathrm{D}t}\\
\frac{\partial \eta}{\partial q} &= \frac{\partial \xi}{\partial q}\nabla\\
\frac{\partial \mu}{\partial q} &= \frac{4\pi R_\odot^3}{M_\odot}\rho\zeta^2\frac{\psi}{\theta}\\
\exp(\xi) &= \exp(\xi_g) + P_{\mathrm{turb}}\\
\gamma &= \ln\rho(\exp(\xi_g), \exp(\eta), X_i)\\
\frac{\partial \lambda}{\partial q} &= \frac{M_\odot}{L_\odot}\left[\epsilon - \frac{\mathrm{D}U}{\mathrm{D}t} + \frac{P}{\rho^2}\frac{\mathrm{D}\rho}{\mathrm{D}t}\right]\frac{\psi}{\theta}\\
\frac{\partial \zeta}{\partial q} &= \frac{\psi}{\theta}\\
\frac{\partial \psi}{\partial q} &= 0\\
\frac{\partial X_i}{\partial t} &= -\frac{2}{3M_\odot\sqrt{\nu}}\frac{\partial F_i}{\partial \mu} + \Psi_i(\xi_t, \eta; \mathcal{X}),\ 1 \le i \le n_{\mathrm{elem}}
\end{aligned}
$$

$$
\left\{
\begin{aligned}
&\text{Conditions limites:}\\
\mu(n,t) &= \mu_{\mathrm{ext}}\left(\lambda(n,t)L_\odot, \zeta(n,t)R_\odot\right),\\
\xi(n,t) &= \ln P_{\mathrm{ext}}\left(\lambda(n,t)L_\odot, \zeta(n,t)R_\odot\right),\\
\eta(n,t) &= \ln T_{\mathrm{ext}}\left(\lambda(n,t)L_\odot, \zeta(n,t)R_\odot\right),\\
\xi_{\mathrm{g}}(n,t) &= \xi(n,t),\\
\gamma(n,t) &= \ln\rho(P_{\mathrm{g}}(n,t), T(n,t), X_i),\\
\zeta(1,t) &= 0,\\
\lambda(1,t) &= 0,\\
\mu(1,t) &= 0.\\
&\text{Conditions initiales:}\\
X_i(q,0) &= X_{i,0},\ i = 1, \ldots, n_{\mathrm{elem}},\\
\xi(q,0) &= \xi_0(q),\\
\eta(q,0) &= \eta_0(q),\\
\gamma(q,0) &= \gamma_0(q),\\
\zeta(q,0) &= \zeta_0(q),\\
\lambda(q,0) &= \lambda_0(q),\\
&1 \le q \le n.
\end{aligned}
\right. \tag{6.32}
$$

We used the notation: $\nu \equiv \mu^{2/3}$. These physical space equations are formed in the routine `static_r` (see Sect. 7.76). Thus, the free boundary condition is, in fact, already included in the equations written with the repartition function and does not entail any fundamental modification of structure.

6.2.5 Node on a RZ/CZ boundary

At each boundary between a convective zone and a radiative zone, the gradient:

$$\nabla \equiv \left(\frac{\mathrm{d}\ln T}{\mathrm{d}\ln P}\right)_t \tag{6.33}$$

is not differentiable. Therefore, the first derivative of temperature is also not differentiable. Between two points of the integration grid, the unknowns are represented by piecewise polynomials. A boundary between a radiative zone and a convective zone must necessarily lie on a mesh node. With Cesam2k20, it suffices to place a grid point better than $\sim 5\%$ of the distance between the nodes immediately surrounding each boundary between a convective zone and a radiative zone; then, there is no risk of approximating the unknown functions by a polynomial piece crossing a discontinuity, as the collocation points are located at a distance from the grid points greater than 5% of the width of the concerned mesh.

To ensure this arrangement, a free parameter of the repartition function is used. From one mesh to another, the variation of the repartition function $Q(\mu, t)$ is constant. The distribution factors a and b define the relative weights of the variables within each mesh; these weights are defined up to a multiplicative factor. So far, this free parameter has been implicitly set to unity. Modifying this value for meshes located in the vicinity of a boundary between a radiative zone and a convective zone will allow the boundary to be pushed into the immediate vicinity of the grid points. We define the step function $\omega(q)$, adjustable from one mesh to another by:

$$Q(\mu, t) = \omega(q) Q_0(\mu, t) \tag{6.34}$$

where $Q_0(\mu, t)$ now denotes the retained repartition function, i.e., the one defined by the relation (6.31). While respecting the constant jump of $Q(\mu, t)$ from one mesh to another, $\omega(q)$ allows adjusting the jump of Q_0 in each mesh, thus the jumps of ξ, η, ζ, λ, and μ so that they correspond to the quantity necessary to reach exactly the boundary. Once the condition:

$$\frac{\mathrm{d}^2 Q}{\mathrm{d}q^2} = \frac{\mathrm{d}\psi}{\mathrm{d}q} = 0 \tag{6.35}$$

is satisfied at all points of the grid, in a given mesh, the variation of the function Q being constant:

$$\frac{\mathrm{d}Q}{\mathrm{d}q} = \psi = \text{constant} = \frac{\mathrm{d}(\omega Q_0)}{\mathrm{d}q} = \omega \frac{\mathrm{d}Q_0}{\mathrm{d}q} \Rightarrow \omega = \frac{\psi}{\frac{\mathrm{d}Q_0}{\mathrm{d}q}} \sim \frac{\psi}{\frac{\Delta Q_0}{\Delta q}} = \frac{\psi}{\Delta Q_0} \tag{6.36}$$

since from one mesh to the next[6] $\Delta q = 1$. For each mesh $[i, i+1], i = 1, \ldots, n-1$, we determine ω_i so that the jump of Q_0:

$$\Delta Q_0 = a(\xi_{i+1} - \xi_i) + b(\mu_{i+1} - \mu_i) \tag{6.37}$$

corresponds to what is necessary for each boundary between a convective zone and a radiative zone to lie on a grid point[7].

Unfortunately, there is no relation allowing the prediction of the movement of a boundary between a convective zone and a radiative zone based on the variations of the local variables, so it is necessary

[6] It is this last relation that defines $\omega(q)$, a step function being not differentiable! For ease of exposition, we preferred to introduce $w(q)$ through (6.34).

[7] Similar relations can be obtained with the distribution factors $\Delta\xi$, $\Delta\eta$, $\Delta\zeta$, $\Delta\lambda$.

to proceed by successive adjustments; as a result, the convergence of the overall iterative process can only be first-order.

At each iteration of the Newton-Raphson method, the boundary between a radiative zone and a convective zone is localized by linear interpolation as a function of the index variable q. Depending on the interpolated position for the boundaries, the involved meshes are enlarged or tightened. Using the obtained value for the constant $\psi(t)$ and the values of the variable Q_0, new values of $\omega(q)$ are calculated for each mesh. An example of such an adjustment is illustrated in Fig. 6.3 (Page 130).

In the `lim_zc` routine (see Sect 7.42) the RZ/CZ boundaries are accurately localized, allowing the determination of the weights to be assigned to each mesh; computational experience shows that the positioning of the boundary is done on a node with an accuracy greater than 1%, if the boundary is well defined. When the RZ/CZ transition is blurred, i.e., when the adiabatic and radiative gradients are close to each other near the boundary, the positioning is not precise (see Sect. 5.3). In these cases, the discontinuity of the gradient's derivative is *small*, and the resulting error is not significant for the structural variables or for the chemical composition, which, with a blurred boundary, can only present a faint discontinuity.

6.3 Restitution of the atmosphere

The light reaching from the stars is the main source of information allowing inference of their structure. A stellar evolution model must describe the transfer of radiation through the atmosphere. Although the latter represents only a tiny part of the star, calculating this transfer is a complex problem that cannot reasonably be solved simultaneously with that of the internal structure. One of the difficulties lies in the fact that it is in the atmosphere that the radiation transitions from the opaque medium of the interior where it is almost isotropic, to the interstellar medium where it is anisotropic. The necessity is to transport the boundary conditions from the transparent medium, where they are defined by observation, to the thick interior medium. The restitution of the atmosphere then consists of constructing a model as accurate as possible, ensuring the transfer of luminous energy and quasi-static equilibrium.

The external boundary conditions of the differential problem of the internal structure concern two of the three thermodynamic variables P, T, and ρ. Cesam2k20 offers the possibility of using two types of methods: the single-layer approximation and the reconstruction of the atmosphere, the latter constituting a boundary value differential problem numerically solved by collocation, (Morel et al., 1994).

6.3.1 The single-layer approximation

The "Solar Model Comparison Project" (Christensen-Dalsgaard, 1988) uses a simplified atmosphere to transport the external boundary conditions into the optically thick medium; it is obtained using two relations deduced from a simple discretization of the atmosphere equations:

$$\left\{\begin{array}{l} \frac{\mathrm{d}P}{\mathrm{d}\tau} = \frac{\mathcal{G}M_\star}{R_\star^2\kappa} - \frac{2\Omega^2 R_\star}{3\kappa}, \\ \frac{\Delta P}{\Delta\tau} \sim \frac{P-P_{\rm ext}}{\tau-0} \\ L_\star = 4\pi R_\star^2 \sigma T^4 \end{array}\right. \Rightarrow \left\{\begin{array}{l} P = \frac{\mathcal{G}M_\star}{R_\star^2\kappa} - \frac{2\Omega^2 R_\star}{3\kappa} \\ L_\star = 4\pi R_\star^2 \sigma T^4 \end{array}\right. \tag{6.38}$$

on the outer layer we then have $T_{\rm ext} = T_{\rm eff}$; σ is the Stefan-Boltzmann constant, $R_\star$, $M_\star$, Ω, and $L_\star$ are respectively the radius, mass, angular velocity, and luminosity of the star. The last two relations are obtained with $\tau = 1$ and $P_{\rm ext} = 0$ which, physically, can be interpreted by saying that the external boundary conditions are written at $\tau = 1$. This formulation is used in the routine `lim_tau1` (see Sect. 7.10.3). The evolution is stopped if the centrifugal acceleration becomes greater than 90% of gravity - which would result in negative pressure.

In the routine lim_gong1 (see Sect. 7.10.3), two factors $\beta = 7.22$ and $\lambda = 6$, respectively affect the equations of pressure and luminosity, their effect being, with an EoS assuming total ionization, to obtain a solar model that resembles the current Sun.

Problem: *With the single-layer approximation, turbulent pressure is ignored in the atmosphere.*

6.3.2 Atmosphere reconstruction

In cases where high precision is required, especially for solar models used in oscillation calculations, it is not possible to ignore the fine structure of the outer layers where oscillation modes reflect. It is necessary to reconstruct this boundary layer as accurately as possible; this is the purpose of the lim_atm routine (see Sect. 7.10.2). In a calculation of internal structure, it is not realistic to treat radiative transfer with all its complexity. The diffusion approximation, gray case, used in internal structure, is justified only for Rosseland optical depths $\tau_{\rm Ross} \geq 20$ Morel et al. (1994), beyond which the radiative flux calculated by the diffusion approximation and the actual flux differ by less than 1%. To avoid explicitly treating the atmosphere, external conditions i.e. those at the junction with the interstellar medium, namely: density or pressure, temperature, and mass, are transported from $\tau \sim 0$ to $\tau = \tau_{\rm Ross}$.

With Cesam2k20, to connect the atmosphere model and the internal structure model, three functions are needed for pressure, temperature, and mass at the junction point $\tau = \tau_{\rm Ross}$ i.e. at the external boundary of the envelope:

$$P_{\rm Ross}(R, L),\ T_{\rm Ross}(R, L),\ M_{\rm Ross}(R, L), \tag{6.39}$$

where $M_{\rm Ross}(R, L) \simeq M_{\rm tot}$ is the mass contained within the outermost layer of the envelope. In the atmosphere, the luminosity L is assumed constant $L = L_\star$. The chemical composition, angular velocity, which may vary over time e.g. due to diffusion and convective mixing, are assumed constant and set to their values at the outermost layer of the envelope. For a given mixing length and chemical composition, the temperature in the atmosphere is given by a relation $T(\tau)$ of the form: $T(\tau_{\rm Ross}, T_{\rm eff}, g)$ depending on the Rosseland optical depth $\tau_{\rm Ross}$, $d\tau = -\kappa\rho dR$, the effective temperature $T_{\rm eff}$, and the gravity g:

$$T_{\rm eff} = \left(\frac{L}{4\pi R_{\rm tot}^2 \sigma}\right)^{\frac{1}{4}}, \quad g = \frac{\mathcal{G}M}{R^2} \cong \frac{\mathcal{G}M_{\rm tot}}{R_{\rm tot}^2}, \tag{6.40}$$

where $R_{\rm tot}$ is the radius at the junction with the interstellar medium (**?**), and κ is the opacity.

This definition of the effective temperature corresponds to the temperature of a blackbody radiating the same energy density as the star, and it is not universal. A second definition, often used, defines an "observable radius" $R_\star$:

$$R_\star = R(\tau = \frac{2}{3}) \quad \text{and} \quad T_{\rm eff} = \left(\frac{L}{4\pi\sigma R_\star^2}\right)^{\frac{1}{4}}. \tag{6.41}$$

Under these conditions, the mass $M_\star < M_{\rm tot}$ is that at the optical depth $\tau = 2/3$: $M_\star = M(\tau = 2/3)$.

In Cesam2k20, a third definition of $T_{\rm eff}$ (Schatzman & Praderie, 1990) is used: the radius of the star $R_\star$ is defined at the optical depth where the temperature is equal to the effective temperature, $R_\star$ is the "bolometric radius". For some $T(\tau)$ relations, the latter two definitions are equivalent. Hereinafter, $\tau_\star$ denotes the optical depth where the radius is that of the star: $R(\tau_\star) = R_\star$.

The chemical composition, opacity, and the chosen definition for the effective temperature must obviously be the same in:

- the atmosphere model calculation program from which the $T(\tau)$ relation is derived,
- the atmosphere part of the internal structure program,

- the upper part of the envelope.

Knowing $R_{\rm Ross}$ and L, we can integrate from $\tau = \tau_{\rm min}$ to $\tau = \tau_{\rm Ross}$ where, at radius $R_{\rm Ross}$, the three quantities are obtained:

$$P_{\rm Ross}(R,L),\ T_{\rm Ross}(R,L),\ M_{\rm Ross}(R,L). \tag{6.42}$$

By using τ as an independent variable, the system of equations to be integrated is:

$$\left\{\begin{array}{rcl}
\frac{{\rm d}P}{{\rm d}\tau} &=& \frac{\mathcal{G}M}{\kappa R^2} - \frac{2\Omega^2 R}{3\kappa} \\
\frac{{\rm d}T}{{\rm d}\tau} &=& \frac{T}{P}\frac{{\rm d}P}{{\rm d}\tau}\nabla \\
\frac{{\rm d}R}{{\rm d}\tau} &=& -\frac{1}{\kappa\rho} \\
\frac{{\rm d}M}{{\rm d}\tau} &=& -\frac{4\pi R^2}{\kappa} \\
\frac{{\rm d}R_\star}{{\rm d}\tau} &=& 0 \\
\frac{{\rm d}\tau_\star}{{\rm d}\tau} &=& 0 \\
P_{\rm gas} &=& P\left[1 - p\frac{a\alpha^2\delta}{8}\frac{\Gamma}{\Gamma+1}(\nabla - \nabla^*_{\rm ad})\right]
\end{array}\right.
\qquad
\left\{\begin{array}{rcl}
\text{boundary conditions:} & & \\
\rho(\tau_{\rm min}) &=& \rho_{\rm ext} \\
\text{or } P(\tau_{\rm min}) &=& g\kappa\tau_{\rm min} \\
P_{\rm gas}(\tau_{\rm min}) &=& P(\tau_{\rm min}) \\
M(\tau_\star) &=& M_\star \\
R(\tau_\star) &=& R_\star \\
T_{\rm eff} &=& \mathcal{T}(\tau_\star, T_{\rm eff}, g_\star) \\
T(\tau_\star) &=& \mathcal{T}(\tau_\star, T_{\rm eff}, g_\star) \\
R(\tau_{\rm Ross}) &=& R_{\rm Ross}
\end{array}\right. \tag{6.43}$$

The two differential equations and the two boundary conditions for $R_\star$ and $\tau_\star$ have been added to solve the differential problem with the free boundary at $\tau = \tau_\star$, see Stoer & Bulirsch (1979, Sect. 7.3.0). They couple $R_\star$ and $\tau_\star$ with the other variables. By coding `lim_ro=.TRUE.` in the data file, at the optically thin limit, defined by $\tau = \tau_{\rm min}$, the boundary condition is on the density fixed at the value $\rho_{\rm ext}$ given by the atmosphere model; with `lim_ro=.FALSE.`, the boundary condition is on the pressure; the relation derived from the single-layer approximation is then used. The pressure condition avoids the constraint on density at an arbitrarily chosen optical depth. In general, $\tau_{\rm min} \equiv 10^{-4}$ is set, which approximately corresponds to the minimum temperature in the solar photosphere, but this is obviously not an absolute rule[8]. At this level, the atmosphere is assumed to be radiative, so that $P_{\rm gas} \equiv P$.

The last equation relating P and $P_{\rm gas}$ is the one satisfied in radiative zones where the efficiency of convection is zero i.e. $\Gamma \equiv 0$.

The boundary conditions are thus written at three levels: $\tau = \tau_{\rm Ross}$, $\tau = \tau_\star(t)$, and $\tau = \tau_{\rm min}$, remembering that $\tau_\star(t)$ may vary over time. Additionally, $\rho_{\rm ext}$ also depends on time, as it depends on gravity $\rho_{\rm ext}(g_\star(t))$.

At time t, the solution of Eqs. (6.43) gives $P_{\rm Ross}$, $T_{\rm Ross}$, and $M_{\rm Ross}$ for given R and L.

Obviously, as far as possible, the $T(\tau)$ relation and the internal structure should be calculated with physics as equivalent as possible, namely: equation of state, opacity, mixing length, chemical composition, etc. Despite many attempts, due to inconsistencies between the physics, data, and parameters used in the atmosphere model on one hand, and in the internal structure on the other hand, it has proved impossible to connect not only the functions (i.e. pressure, temperature, mass) but also their gradients at the boundary between atmosphere and envelope; while the pressure gradient i.e. ${\rm d}P/{\rm d}r = -g\rho$ fits perfectly, the same cannot be said for the temperature gradient, and consequently, the gradient $\nabla \equiv {\rm d}\ln T/{\rm d}lnP$; differences of the order of 0.1 to 0.2 may exist on either side of the junction. For some applications, such differences are prohibitive, and, at the risk of not perfectly verifying the $T(\tau)$ relation, a semi-original numerical method has been developed to also ensure the connection of ∇.

[8] It must be ensured that the equation of state and opacity tables provide realistic values for the physical conditions expected up to $\tau_{\rm min}$.

Problem: *Atmosphere reconstruction is only performed if the centrifugal acceleration is less than 90% of gravity; if not, the evolution is halted.*

6.3.3 Numerical techniques used for the connection of ∇

Given that gravity g is fixed, the $T(\tau)$ relation is assumed to be expressible as: $T^4 = \frac{3}{4}T_{\rm eff}^4 f(\tau)$; using the expression for luminosity, $L = 4\pi R_\star^2 \sigma T_{\rm eff}^4$, it follows:

$$T^4 = \frac{3}{4\pi ac}\frac{L_\star}{R_\star^2}f(\tau), \quad (\sigma = \frac{ac}{4}), \text{ hence } \frac{{\rm d}T}{{\rm d}\tau} = \frac{3}{16\pi ac}\frac{L_\star}{R_\star^2 T^3}\frac{{\rm d}f}{{\rm d}\tau}, \tag{6.44}$$

so that with $d\tau = -\kappa\rho dr$:

$$\begin{aligned}
\nabla &\equiv \frac{{\rm d}\ln T}{{\rm d}\ln P} \\
&= -\frac{P}{T}\frac{{\rm d}T}{{\rm d}\tau}\kappa\rho\frac{{\rm d}r}{{\rm d}P} \\
&= \frac{P\kappa}{T}\frac{{\rm d}T}{{\rm d}\tau}\frac{R^2}{GM} \\
&= \frac{3}{16\pi acG}\frac{P\kappa L_\star}{MT^4}\left(\frac{R}{R_\star}\right)^2\frac{{\rm d}f}{{\rm d}\tau} \\
\nabla &= \nabla_{\rm rad}\frac{L}{L_\star}\left(\frac{R}{R_\star}\right)^2\frac{{\rm d}f}{{\rm d}\tau},
\end{aligned} \tag{6.45}$$

since the radiative gradient has the expression:

$$\nabla_{\rm rad} = \frac{3}{16\pi acG}\frac{P\kappa L}{MT^4}. \tag{6.46}$$

Here $L \equiv L_\star$ and, in the case of the Sun $R/R_\star < 1 \pm 10^{-3}$. In a convective zone, the total flux F is the sum of the radiative flux and the convective flux: $F = F_{\rm rad} + F_{\rm conv}$. *If and only if* the diffusion approximation is valid i.e. at the limit $\tau \gg 1$, the radiative flux $F_{\rm rad}(\tau) = \int_0^\infty F_\nu d\nu$ has the expression:

$$F_{\rm rad} = \frac{4acT^4}{3\kappa\rho H_{\rm p}}\nabla \tag{6.47}$$

with the total flux F:

$$F = \frac{4acT^4}{3\kappa\rho H_{\rm p}}\nabla_{\rm rad} \tag{6.48}$$

then:

$$\lim_{\tau\gg 1}\frac{F_{\rm rad}}{F} = \left(\frac{R}{R_\star}\right)^2\frac{{\rm d}f}{{\rm d}\tau} \sim \frac{{\rm d}f}{{\rm d}\tau}, \tag{6.49}$$

since in the atmosphere, at least for dwarf stars, the ratio $(R/R_\star)^2 \sim 1 \pm 10^{-8}$ can be taken equal to unity.

The numerical technique used for the connection of ∇ is based on the formulation of Eq. (6.45): for the calculation of temperature, instead of explicitly using the $T(\tau)$ relation, the differential equation is integrated:

$$\frac{{\rm d}T}{{\rm d}\tau} = \frac{T}{P}\frac{{\rm d}P}{{\rm d}\tau}\nabla = \frac{T}{P}\frac{{\rm d}P}{{\rm d}\tau}\nabla_{\rm rad}\left(\frac{R}{R_\star}\right)^2\frac{{\rm d}f}{{\rm d}\tau}, \tag{6.50}$$

and a boundary condition on temperature, the latter being written, *for example*, at $\tau_\star$:

$$T(\tau_\star) = T(\tau_\star, T_{\rm eff}, g_\star). \tag{6.51}$$

Thus, apart from a change of variable, the equations *are the same* in the reconstructed atmosphere and in the internal structure.

For purely radiative $T(\tau)$ relations, such as those of Eddington or Hopf, $\lim_{\tau\gg1} \frac{{\rm d}f}{{\rm d}\tau} = 1$. It is noted that then, the total flux is set equal to the radiative flux even if the diffusion approximation is unjustified. For $T(\tau)$ relations incorporating convection, such as those derived from solar atmosphere models, $\lim_{\tau\gg1} \frac{{\rm d}f}{{\rm d}\tau} < 1$ (typically $\frac{{\rm d}f}{{\rm d}\tau} \sim 0.2$ at $\tau = 20$). Therefore, a different treatment will be required depending on whether the $T(\tau)$ relation is purely radiative or not.

6.3.4 Purely radiative $T(\tau)$ law

The $T(\tau)$ law does not contain the convective part of the atmosphere, so it is necessary to use the formalism of the internal structure to calculate the gradient in the convective part, knowing that it will be incorrect as long as the diffusion approximation is not justified i.e. as long as $\tau \leq 20$. In the radiative part, the gradient will have the expression (6.45), in the convective part, its expression will be derived from the theory of convection used (see Sect. 6.9.2).

The continuity of the gradient will be ensured if it is calculated with expressions that have the same limits on both sides of the convective/radiative transition, which is achieved by using, in the formalism of mixing length, a *modified* $\nabla_{\rm rad}$:

$$\nabla^*_{\rm rad} \equiv \nabla_{\rm rad} \frac{{\rm d}f}{{\rm d}\tau}. \tag{6.52}$$

This numerical trick is due to Henyey (Henyey et al., 1965), M. Gabriel & J. Christensen-Dalsgaard, *private communications*. At the limit $\tau \gg 1$, the gradient will thus be exactly that given by the mixing length formalism since, for a purely radiative law, $\lim_{\tau\gg1} \frac{{\rm d}f}{{\rm d}\tau} = 1$.

Purely radiative $T(\tau)$ laws implemented in Cesam2k20:

- `hopf`: Hopf's $T(\tau)$ law, see Sect. 7.10.
- `edding`: Eddington's $T(\tau)$ law, see Sect. 7.10.

6.3.5 Non-purely radiative $T(\tau)$ laws

The previous numerical trick cannot be used since $\lim_{\tau\gg1} {\rm d}f/{\rm d}\tau < 1$ and, moreover, using a non-purely radiative $T(\tau)$ law would lose its interest since the description of the convective part at optical depths $\tau \sim 1$ would be done with a convection theory that uses a radiative flux expression derived from the diffusion approximation. In this case, to ensure the continuity of the gradient, the numerical trick used consists of a linear interpolation, as a function of optical depth, between the value of the gradient derived from the $T(\tau)$ law:

$$\nabla = \nabla_{\rm rad} \left(\frac{R}{R_\star}\right)^2 \frac{{\rm d}f}{{\rm d}\tau} \tag{6.53}$$

and that $\nabla_{\rm mix}$ calculated by the mixing length theory as described in Sect. 6.9.2. Thus, for optical depths $\tau \leq 1$, the gradient is close to that derived from the $T(\tau)$ law, while it tends towards that of the mixing length for optical depths $\tau \geq \tau_{\rm rac}$. The linear interpolation used is given by:

$$\nabla = (1-x)\nabla_{\rm rad} \left(\frac{R}{R_\star}\right)^2 \frac{{\rm d}f}{{\rm d}\tau} + x\nabla_{\rm mix}, \quad \text{with} \quad x = \frac{\tau - 1}{\tau_{\rm rac} - 1}. \tag{6.54}$$

For intermediate optical depths $\tau \sim 1$, this numerical trick allows to partially restore the atmosphere model with convection, while ensuring, deeper, the connection of the gradients despite the inconsistencies of the physics.

Non-purely radiative $T(\tau)$ laws implemented in Cesam2k20:

- `k5750`: Solar $T(\tau)$ law $T_{\rm eff} = 5750$ K calculated by C. Van't Veer with Atlas 9 from Kurucz (see Sect. 7.10).
- `k5777`: Solar $T(\tau)$ law $T_{\rm eff} = 5777$ K calculated by C. Van't Veer with Atlas 12 from Kurucz (see Sect. 7.10).
- `roger`: $T(\tau)$ laws at $[{\rm Fe/H}] \in [0.0, -0.2, -0.5, -1.0]$, calculations with Atlas 12 from Kurucz, interpolations by R. Cayrel. Uses the table `fesh00.data` from the directory `SUN_STAR_DATA/ATM/Ttau/roger` of the Cesam2k20 source; before use, indicate in the subroutine the access path of the table (see Sect. **??**).

6.3.6 Numerical resolution

To numerically solve the differential problem defined by the system (6.43), we impose (in the `lim_atm` routine) a *fixed* index $n_\star = 3/4 n_{\rm a}$ at the outer boundary $\tau = \tau_\star(t)$ among the $n_{\rm a}$ grid points in the atmosphere. To do this, we use the *piecewise linear bijection*: $\tau \mapsto \omega$ from $[\tau_{\rm rac}, \tau_{\rm min}]$ to $[1, n_{\rm a}]$ defined by:

$$\ln \tau \mapsto \varphi(\omega) = \begin{cases} \ln \tau_{\rm rac} + (\omega - 1)\Delta\varphi^+, & \text{if } \omega \in [1, n_\star]; \\ \ln \tau_{\rm min} + (\omega - n_{\rm a})\Delta\varphi^-, & \text{otherwise.} \end{cases} \tag{6.55}$$

The two values of the slope $\Delta\varphi^+$ and $\Delta\varphi^-$ are defined by:

$$\Delta\varphi^+ \equiv \frac{\ln \tau_\star - \ln \tau_{\rm min}}{n_\star - n_{\rm a}}, \quad \Delta\varphi^- \equiv \frac{\ln \tau_\star - \ln \tau_{\rm rac}}{n_\star - 1}; \tag{6.56}$$

thus, we have:

- $\omega = 1$, at the bottom of the atmosphere where $\tau = \tau_{\rm rac}$,
- $\omega = n_\star$, on the inner free boundary at $\tau = \tau_\star$,
- $\omega = n_{\rm a}$, at the outer boundary where $\tau = \tau_{\rm min}$.

We work with a constant grid step, which is an advantage of the method. For numerical integration, we use the variables: $\xi = \ln P$, $\xi_{\rm g} = \ln P_{\rm gas}$, $\eta = \ln T$, $\zeta_{\rm a} = R/R_\odot$, $\mu_{\rm a} = M/M_\odot$, $\zeta_\star = R_\star/R_\odot$ and $\varphi_\star = \ln \tau_\star$, so that the system (6.43) becomes:

$$\begin{cases} \frac{{\rm d}\xi}{{\rm d}\omega} & = \left(\frac{GM_\odot}{R_\odot^2} \frac{\mu_{\rm a}}{\zeta_{\rm a}^2} - \frac{2R_\odot \Omega^2 \zeta_{\rm a}}{3} \right) \frac{\Delta\varphi^\pm}{\kappa} \exp(\varphi - \xi) \\ \frac{{\rm d}\eta}{{\rm d}\omega} & = \frac{{\rm d}\xi}{{\rm d}\omega} \nabla \\ \frac{{\rm d}\zeta_{\rm a}}{{\rm d}\omega} & = -\frac{1}{R_\odot} \frac{\Delta\varphi^\pm \exp(\varphi)}{\kappa\rho} \\ \frac{{\rm d}\zeta_\star}{{\rm d}\omega} & = 0 \\ \frac{{\rm d}\mu_{\rm a}}{{\rm d}\omega} & = -\frac{4\pi R_\odot^2}{M_\odot} \frac{\Delta\varphi^\pm}{\kappa} \exp(\varphi) \zeta_{\rm a}^2 \\ \frac{{\rm d}\varphi_\star}{{\rm d}\omega} & = 0 \\ \frac{{\rm d}\varphi}{{\rm d}\omega} & = \Delta\varphi^\pm \\ 0 & = \exp(\xi) \left[1 - \frac{a\alpha^2\delta}{8} \frac{\gamma}{\gamma+1} (\nabla - \nabla^*_{\rm ad}) \right] - \exp(\xi_{\rm g}) \end{cases} \tag{6.57}$$

with $\Delta\varphi^{\pm} = \Delta\varphi^{+}$ (resp. $\Delta\varphi^{-}$) if $\omega \in [1, n_\star]$ (resp. $\omega \in [n_\star, n_\mathrm{a}]$).

The boundary conditions are then:

$$\begin{cases} \zeta_\mathrm{a}(1) & = & \zeta_\mathrm{rac} \\ \mu_\mathrm{a}(n_\star) & = & M_\star/M_\odot \\ \zeta_\star(n_\star) & = & R_\star/R_\odot \\ \eta(n_\star) & = & \ln \mathcal{T}(\exp(\varphi_\star), T_\mathrm{eff}, g_\star) \\ \eta(n_\star) & = & \ln T_\mathrm{eff} \\ \varphi(n_\mathrm{a}) & = & \ln \tau_\mathrm{min} \\ \rho(n_\mathrm{a}) & = & \rho_\mathrm{ext} \\ \xi(n_\mathrm{a}) & = & \xi_\mathrm{g}(n_\mathrm{a}) \end{cases} \tag{6.58}$$

with:

$$T_\mathrm{eff}^4 = \frac{L_\odot\sqrt{\lambda_\mathrm{rac}^3}}{4\pi R_\odot^2 \sigma \zeta_\star^2}, \quad g = \frac{GM_\odot \mu_\mathrm{a}}{R_\odot^2 \zeta_\mathrm{a}^2}. \tag{6.59}$$

The numerical solution of this nonlinear differential problem uses the spline-collocation method. The iterative process is initialized, during the first calculation, by a solar atmosphere model and, during the evolution, by the atmosphere model from the previous time step.

Problem: *Due to the discontinuity of the derivative of the piecewise linear bijection (see Sect. 5.3), at the point $\tau = \tau_\star$, convergence difficulties seem to occur when $\tau_\star$ tries to cross the discontinuity, a large number of iterations is then necessary. In fact, a property of the spline-collocation method allows to effectively reach the solution.*

Typically, we use $n_\mathrm{a} = 30$ to $n_\mathrm{a} = 100$ grid points for the interval $[\tau_\mathrm{rac}, \tau_\mathrm{min}]$. With third-order B-splines, there are 59 to 199 integration points where Eqs. (6.57) are satisfied, taking into account superconvergence, the scheme order is 4.

These equations are solved in the spline space by the `coll_atm` routine, (see Sect. 7.9.2), using the coefficients calculated in the physical space by the `eqatm` routine (see Sect. 7.10.1). The management and interface between the two spaces being ensured by the `lim_atm` routine, (see Sect. 7.10.2). With $T(\tau)$ laws derived from sophisticated atmosphere models, the optical depth $\tau_\star$ varies with effective temperature and gravity, the implicit equation $T(\tau_\star) = T_\mathrm{eff}$ is then solved in the `taueff` routine, (see Sect. 7.78).

In the resolution program, the index 1 corresponds to the connection at $\tau = \tau_\mathrm{rac}$ and $R = R_\mathrm{rac}$, the index n_a to $\tau = \tau_\mathrm{min}$. The external density ρ_ext at $\tau = \tau_\mathrm{min}$ depends on the $T(\tau)$ law used. The optical depth $\tau = \tau_\mathrm{rac}$ of the bottom of the atmosphere is an external parameter introduced in the NAMELIST `NL_ATM` of the data file: τ_rac = `tau_max`.

The solution consists of $\xi_\mathrm{rac}(\zeta_\mathrm{rac}, \lambda_\mathrm{rac})$, $\eta_\mathrm{rac}(\zeta_\mathrm{rac}, \lambda_\mathrm{rac})$, $\mu_\mathrm{rac}(\zeta_\mathrm{rac}, \lambda_\mathrm{rac})$; the derivatives with respect to η_rac and λ_rac are calculated numerically.

6.4 Temporal evolution of gravitational energy

The gravitational energy plays a predominant role in the regions of the star where it replaces nuclear energy, for example in the core after the main sequence or during the Hayashi phase. It is this equation, reflecting the assumption of quasi-static equilibrium, that avoids a hydrodynamic treatment of stellar evolution. It is not justified in all phases of evolution.

6.4.1 Kippenhahn's approximation

Although it is formally incorrect (Cox & Giuli, 1968; Strittmatter et al., 1970), at a Lagrangian abscissa where the temporal variation of the number of free particles is not zero, for example in a convective

core, the approximation by Kippenhahn et al. (1967) is given by:

$$\epsilon_{\rm G} \sim T\left(\frac{\partial S}{\partial t}\right) \tag{6.60}$$

where S is the specific entropy. This approximation allows the gravitational energy to be expressed explicitly in terms of variations in pressure and temperature:

$$\epsilon_{\rm G} \sim T\left(\frac{\partial S}{\partial t}\right) = c_{\rm P}\frac{\partial T}{\partial t} - \frac{\delta}{\rho}\frac{\partial P}{\partial t} \tag{6.61}$$

and is easy to use. For stars of mass ($M \geq 1.2M_{\odot}$), the extension of the convective core varies during evolution, creating a discontinuity in chemical composition and therefore in density along the temporal axis; a discontinuity that needs to be taken into account when evaluating the temporal derivative of density in Eq. (6.63). Kippenhahn's approximation, which does not explicitly use the derivative of a discontinuous function, allows us to circumvent the difficulty and to let the numerical methods handle the error.

6.4.2 Discretization

The variation of gravitational energy:

$$\epsilon_{\rm G} = \left(\frac{\partial U}{\partial t}\right)_{\mu} + P\left(\frac{\partial V}{\partial t}\right)_{\mu} \tag{6.62}$$

where U is the specific internal energy, $V = 1/\rho$ is the specific volume, and μ is the Lagrangian spatial variable, is approximated in Cesam2k20 by an implicit finite difference scheme:

$$\epsilon_{\rm G} \approx \frac{U^{t+\Delta t} - U^t}{\Delta t} - \frac{P^{t+\Delta t}}{(\rho^{t+\Delta t})^2}\frac{\rho^{t+\Delta t} - \rho^t}{\Delta t} \tag{6.63}$$

where the quantities U^t and ρ^t at time t are calculated using the equation of state with pressure and temperature values obtained by interpolation at μ, i.e., constant mass, from the previous solution:

$$\mu \hookrightarrow q \hookrightarrow P^t, T^t \tag{6.64}$$

and, for the chemical composition X, by direct interpolation with respect to μ, of order `mc`, the order of B-spline interpolation of the chemical composition.

With the Kippenhahn approximation, the fully implicit finite difference formulation is written as:

$$\epsilon_{\rm G} \approx c_{\rm P}^{t+\Delta t}\frac{T^{t+\Delta t} - T^t}{\Delta t} - \frac{\delta^{t+\Delta t}}{\rho^{t+\Delta t}}\frac{P_{\rm gas}^{t+\Delta t} - P_{\rm gas}^t}{\Delta t} \tag{6.65}$$

These equations are formed in the routines `static_m` and `static_r` (see Sect. **??**).

6.4.3 Initialization

For the calculation of the pre-main sequence (PMS), it is necessary to have a model in quasi-static equilibrium where the only source of energy is of gravitational origin. The initial model is obtained using a method due to Iben (1965), which has been adapted to Cesam2k20 in collaboration with A. Baglin. At the beginning of the PMS, the star is completely convective and thus isentropic, except perhaps in an external superadiabatic region. Under these conditions, the chemical composition is spatially and temporally *constant*, and the energy equation is written as:

$$\frac{\partial L}{\partial m} = \epsilon_G = -T\frac{\partial S}{\partial t} = cT \tag{6.66}$$

where c is the "*contraction constant*" that characterizes a quasi-static equilibrium model, fully convective and without nuclear reactions. Providing the contraction constant c allows solving the system of equations at time $t + dt$ without explicitly determining the quantity $T\partial S/\partial t$.

The Iben method consists of constructing a first model (index 1) with an initial value c_1 for c. Then, a second model (index 2) with a nearby value c_2 is constructed. Cesam2k20 uses $c_2 = 1.1 \times c_1$. If we assume that these two models belong to the same evolutionary sequence, the radiated energy during the time interval Δt separating them is then equal to the change in gravitational energy, so:

$$\frac{L_1 + L_2}{2}\Delta t \sim \left(\frac{GM^2}{R_2} - \frac{GM^2}{R_1}\right) \Rightarrow \Delta t \sim 2GM^2 \frac{R_1 - R_2}{(L_1 + L_2)R_1R_2}. \tag{6.67}$$

We can then start the evolution calculation using this value of Δt and model 2 as the initial model. A typical value for the constant is $c = 0.02$, so the central temperature of the initial model is around 100000K; we obtain 500000K (resp. 1000000K) with $c = 0.0005$ (resp. $c = 0.00008$). These orders of magnitude depend little on the mass of the model. During the course of the calculation, Cesam2k20 asks for the value of the contraction constant c to use. At each step of determining the initial PMS model, it is possible to check the values of the central temperature, luminosity, and total radius in order to modify c until obtaining the desired model. Initialization for the calculation of the first model can be done either from a PMS model calculated with nearby parameters, in a binary file with extension `_B.pms`, or with one of the ASCII files of PMS model, such as `2d-2.pms`, `5d-4.pms`, `8d-5.pms` in the `EXPLOIT` sub-directory. Once the initial PMS model is obtained, it is written for further initialization in the binary file `mon_modele_B.pms`. The initial gravitational energy of the PMS is calculated by the `iben` routine (see Sect. 7.34).

6.5 Evolution of chemical composition without diffusion

The temporal evolution of chemical composition poses particular difficulties because, even on the main sequence where the nonlinear terms of the equations do not yet play a determining role, the evolution equations are delicate to integrate. To avoid degrading the precision order of the quasi-static equilibrium equations resolution, the scheme used for the initial value problem should be of as high an order as possible. Even without diffusion, analysis shows that it is difficult to use an integration formula of order higher than 2 that satisfies the following constraints imposed by the physics of stellar evolution:

- Coexistence of very different evolution time scales. For example, deuterium has a characteristic evolution time of the order of a year, while that of hydrogen is of the order of a hundred million years. Such a differential problem is referred to as "stiff".
- Presence of convective zones that mix chemical species. Since it is desirable to use the same integration formula at every point of the model, one can only use one- or two-step schemes. Beyond that, the logic of the algorithm managing convective mixing and boundary movement becomes very complicated.
- Conservation of charge, number of nucleons, the integration formula must therefore conserve any linear combination of abundances, it will be referred to as "conservative" thereafter.
- Abundances are positive quantities.

Since the problem of the temporal evolution of chemical composition is a stiff differential problem, in order to justify the choices made in Cesam2k20, some indications concerning the numerical resolution of this type of problem are given; for more information, refer to Hairer & Wanner (1996).

6.5.1 Stiff problem

Numerically integrating a Cauchy differential problem:

$$y'(x) = f(x, y), \quad \text{for } x \in [x_0, x_N], \quad \text{and } y(x_0) = y_0 \tag{6.68}$$

involves choosing a step $h > 0$ and, step by step, starting from the initial value y_0, obtaining the numerical solution using a relation of the form: $y_n = y_{n-1} + h\phi$, where the increment function ϕ depends more or less complicatedly on the solution obtained at points $x_i = x_0 + ih$, $0 \leq i \leq n$, $n \geq 1$. Two concepts emerge from this process: *convergence* and *stability*. A formula is said to be convergent if, for a fixed $x \in [x_0, x_N]$ ($x = x_0 + nh$, $h = (x - x_0)/n$, $n \leq N$), the limit as $h \to 0$ of the numerical solution coincides with the exact solution of the differential problem, that is:

$$\lim_{n\to\infty}(x_0 + nh) = x_N \Rightarrow \lim_{n\to\infty} y_n = y(x_N) \tag{6.69}$$

It is shown that most classical formulas are convergent. Stability is related to the behavior of the numerical solution at infinity. An integration formula is stable if, for a fixed step h:

$$\lim_{n\to\infty}(x_0 + nh) = \infty \Rightarrow \lim_{n\to\infty} y_n = y(\infty) \tag{6.70}$$

Instability is due to the accumulation of truncation errors. If these errors increase less rapidly than the solution, the formula is stable; otherwise, it is unstable.

Stability depends not only on the integration formula but also on the differential problem posed. The study can only be carried out for simple equations. Thus, it is necessary to restrict the definition: for stability to occur, truncation errors must not grow exponentially at infinity; for this reason, the study of stability is reduced to that of the numerical solution of the standard problem: $y'(x) = \lambda y(x)$, $\lambda \in C$. With h fixed, the formula is stable if the numerical solution tends to zero as n tends to infinity. It is then said that the formula is A-stable. Furthermore, it will be called A-positive if, in the domain where it is A-stable and $\lambda \in C$, with $y_0 > 0$, as n tends to infinity, the numerical solution tends to zero while remaining positive.

In general, implicit formulas are A-stable in large regions of the complex plane, but few of them are A-positive.

With a stiff differential problem in which the evolution time scales of the various variables are very different, the difficulty is twofold:

- Obtain a stable solution for all variables with an acceptable time step, i.e., without following the smallest time scale.
- Integrate the significant variables with sufficient precision.

For stellar evolution, unless detailed tracking of the chemical species with the smallest characteristic time scale is desired, a stiff problem formula must be used. However, it should be noted that not all variables will be integrated with the same precision; the role of the stiff problem formula is only to prevent the exponential growth of truncation errors affecting the variables with the smallest time scales. Furthermore, the analysis of the behavior at infinity of the formula can only be done on the standard one-dimensional problem. For a system as complex as the temporal evolution of chemical composition, the theory is not established. When transposing a result established for one dimension to multiple dimensions, difficulties can arise in some cases. In these cases, the simplest remedy is to decrease the time step.

6.5.2 Summary of constraints

For the numerical integration of the system of differential equations governing the temporal evolution of chemical composition, it is necessary to use a formula that:

- Is conservative, i.e., conserves any linear combination of abundances $\sum_i \alpha_i X_i^t = \mathrm{cte}$.
- Is stable or, at least, A-stable, i.e., truncation errors relative to variables with the shortest evolution time scale do not increase exponentially when the time step is large compared to this time scale.
- Is positive.
- Is simple and precise.

The first condition is fulfilled for most classical schemes: Runge-Kutta or related-step schemes, for which the increment function depends linearly on the estimated temporal derivatives at various points. The condition of A-stability is met if an implicit scheme is used. Furthermore, it is desirable to use the same integration scheme at all points in the model.

Unfortunately, the fact that the boundaries of convective zones move over time and the necessity of convective mixing eliminate part of the interest of related-step schemes which have interesting stability properties for stiff problems, e.g., BDF schemes. Similarly, it will only be possible to use Runge-Kutta schemes of order higher than 1 if convective mixing is not taken into account for intermediate models, such as with the formula of **?**.

When microscopic diffusion is omitted, Cesam2k20 uses an implicit Runge-Kutta type LobattoIIIC integration scheme for stiff problems.

6.5.3 The IRK Lobatto IIIC formulas

In a radiative zone, without diffusion, the system of equations governing the temporal evolution of chemical composition can be formally written as:

$$\frac{\partial x_i}{\partial t} = \Psi(T, \rho, \mathcal{X}), \quad 1 \leq i \leq n_{\mathrm{elem}}. \tag{6.71}$$

In the following, we describe the use of the IRK LobattoIIIC formula. For simplicity, we consider only a single chemical species. We denote x_0 (resp. x) the abundance of this element at time t (resp. $t + dt$). With the coefficients from Table 6.1, the IRK LobattoIIIC scheme is:

$$\begin{aligned} x_i &= x_0 + \mathrm{d}t \sum_{j=1}^{s} a_{ij} \Psi(T_j, \rho_j, x_j), \quad i = 1, \ldots, s \\ x &= x_0 + \mathrm{d}t \sum_{j=1}^{s} b_j \Psi(T_j, \rho_j, x_j) \end{aligned} \tag{6.72}$$

where T_j, ρ_j, and x_j are, respectively, the temperature, density, and abundance at intermediate times, $t + c_j dt$, $j = 1, \ldots, s$; s is the number of steps of the IRK LobattoIIIC formula.

However, this elementary formulation of a Runge-Kutta scheme does not give satisfactory results for the problem of chemical species evolution, as significantly negative abundance values can occur; the cause is the multiplication of numerical errors by Lipschitz constants that have high values, e.g., 10^{18}. This difficulty is overcome by writing the first equation (6.72) for the differences $z_i = x_i - x_0$ from the initial quantities (see Hairer & Wanner 1996, Sect. IV.8):

$$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_s \end{pmatrix} = \mathrm{d}t \mathcal{A} \begin{pmatrix} \Psi(T_1, \rho_1, x_0 + z_1) \\ \Psi(T_2, \rho_2, x_0 + z_2) \\ \vdots \\ \Psi(T_s, \rho_s, x_0 + z_s) \end{pmatrix}. \tag{6.73}$$

Table 6.1: Coefficients of (6.72) for IRK Lobatto IIIC formulas of order $p = 1, 2, 4$ with $s = 1, 2, 3$ steps. The usual notation is used, the $c_i,\ i = 1, \ldots, s$ are elements of the first column, the b_i those of the last row, the other coefficients are the a_{ij} of the matrix characterizing the Runge-Kutta formula.

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

The matrix $\mathcal{A} \equiv (a_{i,j})_{i,j=1}^{s}$ of any IRK Lobatto IIIC formula being invertible, noting $(d_1, d_2, \ldots, d_s) \equiv (b_1, b_2, \ldots, b_s)\mathcal{A}^{-1}$, the second equation (6.72) becomes simply[9]:

$$x = x_0 + z_s. \tag{6.74}$$

6.5.4 Mixing of chemical elements without diffusion

In a convective zone (CZ), when the timescale of convection is small compared to that of nuclear reactions, the chemical composition is homogenized by convective movements. For isotope i, the destruction rate Ψ_i is no longer the local quantity $\Psi_i(T, \rho, \mathcal{X})$ but rather a quantity averaged over the entire CZ. Thus, in a CZ, Equation (6.71) must be replaced by:

$$\frac{\mathrm{d}\overline{x_i}}{\mathrm{d}t} = \frac{\int_{CZ} \Psi_i(\rho, T, \bar{\mathcal{X}}, t)\mathrm{d}m}{\int_{CZ} \mathrm{d}m} = \frac{\int_{CZ} \Psi_i(\rho, T, \bar{\mathcal{X}}, t)\sqrt{\nu}\mathrm{d}\nu}{\int_{CZ} \sqrt{\nu}\mathrm{d}\nu}, \quad \nu = \left(\frac{m}{M_\odot}\right)^{2/3} \tag{6.75}$$

where $\bar{\mathcal{X}}$ denotes the average abundance over the mixed zone (MZ), which obviously complicates the situation.

As recalled earlier, we assume that in a CZ, the abundance of each isotope is constant. To revisit the example of a single element, let $\bar{x}$ denote its value in a CZ, and at each time step $t + c_j\mathrm{d}t,\ j = 1, \ldots, s$ of the Runge-Kutta integration, let $\overline{x_j}$ denote its average value. We estimate the integrals in (6.75) using the midpoint formula, so that:

$$\overline{x_j} = \sum_{k \in \mathrm{MZ}} x_{j,k+1/2} \Delta\nu_{k+1/2}, \tag{6.76}$$

where the sum extends over all layers in the CZ, $x_{0,k+1/2}$ is the abundance at the midpoint, $\nu_{k+1/2} \equiv (\nu_{k+1} + \nu_k)/2$, and:

$$\Delta\nu_{k+1/2} \equiv \frac{(\nu_{k+1} - \nu_k)\sqrt{\nu_{k+1/2}}}{\sum_{j \in \mathrm{MZ}} (\nu_{j+1} - \nu_j)\sqrt{\nu_{j+1/2}}}. \tag{6.77}$$

Equation 6.73 becomes:

$$\begin{pmatrix} \bar{z}_1 \\ \bar{z}_2 \\ \vdots \\ \bar{z}_s \end{pmatrix} = \mathrm{d}t\mathcal{A} \begin{pmatrix} \sum_k \Psi\left(T_{1,k+\frac{1}{2}}, \rho_{1,k+\frac{1}{2}}, \overline{x_0} + \overline{z_1}\right) \Delta\nu_{k+\frac{1}{2}} \\ \sum_k \Psi\left(T_{2,k+\frac{1}{2}}, \rho_{2,k+\frac{1}{2}}, \overline{x_0} + \overline{z_2}\right) \Delta\nu_{k+\frac{1}{2}} \\ \vdots \\ \sum_k \Psi\left(T_{s,k+\frac{1}{2}}, \rho_{s,k+\frac{1}{2}}, \overline{x_0} + \overline{z_s}\right) \Delta\nu_{k+\frac{1}{2}} \end{pmatrix} \tag{6.78}$$

[9] This particularity results from the fact that the last row of A is identical to the vector b, e.g., all IRK Lobatto IIIC formulas are built with this property.

where $T_{j,k+\frac{1}{2}}$ (resp. $\rho_{j,k+\frac{1}{2}}$) is the temperature (resp. density) at the intermediate time $t + c_j dt$; $\bar{x}$ is written as:

$$\overline{x} = \overline{x_0} + \overline{z_s}. \tag{6.79}$$

Formally, this formulation is also valid at a point in a radiative zone, with the sums involving only a single value of the index k. To simplify, the value of $\overline{x_0}$ is obtained by summing (6.76) over the mass interval corresponding to the CZ at time $t + dt$, which may not *exactly* coincide with the mass interval at time t. The resulting imprecision is small if the CZ moves little; however, in the case of appearance or disappearance, the error generated can be significant. Taking into account these subtleties would entail great complications in the algorithms, as it would be necessary to explicitly introduce the movement of the CZ into Equation (6.78).

Extending to multiple isotopes presents only fairly technical algorithmic difficulties. With the approximation of integrals (6.76) over the CZ using the midpoint formula, the numerical precision is only second order; a two-point Gauss integral formula would achieve third-order accuracy[10]. In the solar case, the error remains limited because thermonuclear reactions are inefficient in the convective zone; for stellar cases, inaccuracies in theory and physical data far outweigh those generated by a low-order integration.

In the `rk_imps` routine (see Sect. 7.7.1), Equations (6.73) to (6.79) are iteratively solved using the Newton-Raphson scheme[11], with a Jacobian recalculated for each iteration; the implicit method is initialized with zero initial values for z_i; convergence is assumed once the corrections on z_j are, in relative values, less than 10^{-8}; most of the time, the algorithm converges in less than 5 iterations.

The `rk_imps` routine is called by `evol`, (see Sect. 7.3.1.3), which manages the temporal evolution of the chemical composition.

6.5.5 Conservation of the number of nucleons

For each isotope indexed by i, the evolution equations of the chemical composition take the general form:

$$\frac{\mathrm{d}X_i}{\mathrm{d}t} = \dot{X}_i = \sum_{j=1}^{m} a^i_j X_j + \sum_{\substack{j=1 \\ k\geq j}}^{m} a^i_{jk} X_j X_k + \sum_{\substack{j=1 \\ l\geq k\geq j}}^{m} a^i_{jkl} X_j X_k X_l \tag{6.80}$$

The abundances X_i of the m different chemical species satisfy linear conservation relations (number of nucleons, charges) of the form:

$$\sum_{i=1}^{m} \gamma_i X_i = \mathrm{cte} \Rightarrow \frac{\partial}{\partial t} \sum_{i=1}^{m} \gamma_i X_i = 0 = \sum_{i=1}^{m} \gamma_i \dot{X}_i. \tag{6.81}$$

This last relation also serves as an analytical test to verify that there are no errors in the evolution formalism.

A linear integration algorithm perfectly conserves such linear combinations of X_j because:

$$\begin{aligned} \sum_{j=1}^{m} \gamma_j X_{j,n} &= \sum_{j=1}^{m} \gamma_j \sum_{i=1}^{k} \alpha_i X_{j,n-i} + \sum_{j=1}^{m} \gamma_j h \sum_{i=0}^{k} \beta_i \dot{X}_{j,n-i} \\ &= \mathrm{cte} + 0. \end{aligned} \tag{6.82}$$

[10] This would only add further algorithmic complication.

[11] Due to stiffness, the Lipschitz constants are large, and a predictor-corrector scheme is inefficient.

6.5.6 Conservation of baryons and charge

At time t, for a number of isotopes $n_{\rm elem} > 1$, the expressions for the conservation of the baryon number and charge take the general form of a linear relationship between the abundances, with constant coefficients α_i:

$$\sum_{i=1}^{n_{\rm elem}} \alpha_i x_i^t = cte, \tag{6.83}$$

thus:

$$\frac{\partial}{\partial t} \sum_{i=1}^{n_{\rm elem}} \alpha_i x_i = 0 \Longrightarrow \sum_{i=1}^{n_{\rm elem}} \alpha_i \Psi_i = 0. \tag{6.84}$$

Taking into account the linearity of (6.73) and (6.78), with obvious notations:

$$\sum_{i=1}^{n_{\rm elem}} \alpha_i z_i = 0, \tag{6.85}$$

thus:

$$\sum_{i=1}^{n_{\rm elem}} \alpha_i x_i^{t+dt} = \sum_{i=1}^{n_{\rm elem}} \alpha_i x_i^t + \sum_{i=1}^{n_{\rm elem}} \alpha_i z_i = cte; \tag{6.86}$$

in particular, with the LobattoIIIC IRK scheme, the number of baryons and charges is conserved at the level of:

- the closure of the Newton-Raphson scheme,
- the order of the integration scheme used for convective mixing,
- rounding errors.

Numerical tests have shown, in the case of the Sun, that over a time step, the number of baryons was conserved to better than 10^{-13} in relative value (Morel, 1997).

6.5.7 Normalization of the sum of abundances

Although the algorithms used for the numerical resolution of the time evolution equations of chemical species ensure most conservation relationships, they can only do so with the required accuracy. To limit the consequences of these approximations, normalization of the sum of abundances per unit mass is performed to ensure the relation $X + Y + Z = 1$. Normalization is performed in the `evol` routine (see Sect. 7.31.3) at the end of the temporal integration of the chemical composition.

The abundance per mass of the chemical element is written as $X_i = x_i a_i$ where x_i is the mole abundance and a_i is the atomic mass of the element. Thus, we have: $X + Y + Z = \sum_i x_i \nu_i$. With the development on the basis of B-splines of order m, at the abscissa point ν, $\nu_{l-1} \leq \nu < \nu_l$, the mole abundance is written as:

$$x_i = \sum_{j=l-m+1}^{l} x_{ij} N_j^m(\nu). \tag{6.87}$$

and from there:

$$\sum_i X_i = \sum_i a_i \sum_{j=l-m+1}^{l} x_{ij} N_j^m(\nu) = \sum_{j=l-m+1}^{l} N_j^m(\nu) \sum_i a_i x_{ij}. \tag{6.88}$$

We obtain a normalization of the abundances per unit mass X_i by setting:

$$X_i = x'_i a_i, \quad x'_i = \sum_{j=l-m+1}^{l} x'_{ij} N_j^m(\nu), \quad x'_{ij} = \frac{x_{ij}}{\eta_j}, \quad \eta_j = \sum_i a_i x_{ij}. \tag{6.89}$$

Since the sum of the B-splines at any point ν is equal to one, we obtain the desired normalization:

$$X + Y + Z = \sum_{j=l-m+1}^{l} N_j^m(\nu) \sum_i a_i x'_{ij} = \sum_{j=l-m+1}^{l} N_j^m(\nu) \sum_i a_i \frac{x_{ij}}{\eta_j} \tag{6.90}$$

$$= \sum_{j=l-m+1}^{l} N_j^m(\nu) \frac{1}{\eta_j} \sum_i a_i x_{ij} = \sum_{j=l-m+1}^{l} N_j^m(\nu) = 1. \tag{6.91}$$

6.5.8 Estimation of integration accuracy

Despite many attempts, it was not possible to use a valid numerical method to estimate integration accuracy, mainly due to the excessive computational cost of these methods; in Cesam2k20, accuracy is ensured by limiting the relative temporal variation of the abundances of the elements that the user wishes to control.

6.6 Evolution of angular momentum without diffusion

Cesam2k20 considers only one radial dimension; for rotating models, the average value of the centrifugal acceleration on the spherical cap of radius r is taken into account.

Given the assumption of sphericity, the moment of inertia $\mathrm{d}\mathcal{I}$ of the elementary spherical volume $\mathrm{d}V = r\sin\theta \mathrm{d}r\mathrm{d}\theta\mathrm{d}\phi$ at the point with spherical coordinates (r, θ, ϕ) is:

$$\mathrm{d}\mathcal{I} = r\sin\theta\rho r\mathrm{d}r\mathrm{d}\theta\mathrm{d}\phi. \tag{6.92}$$

The moment of inertia per unit mass of the spherical cap is written as:

$$\frac{\mathrm{d}I}{\mathrm{d}m} = \frac{\int_0^{2\pi} \mathrm{d}\mathcal{I}\mathrm{d}\phi}{\mathrm{d}m} = \frac{2\pi r^4 \rho \mathrm{d}r}{4\pi r^2 \rho \mathrm{d}r} \int_0^{\pi} \sin^2\theta \mathrm{d}\theta = \frac{2}{3} r^2. \tag{6.93}$$

The specific angular momentum, i.e., the angular momentum per unit mass, is then:

$$\mathcal{M}_\Omega(t) = \frac{2}{3} r^2 \Omega(t). \tag{6.94}$$

Here, $\Omega(t)$ is the average angular velocity of the spherical cap at time t.

The assumption of *local conservation of specific angular momentum* is written as:

$$\frac{\mathrm{D}\mathcal{M}_\Omega}{\mathrm{D}t} = \frac{\mathrm{D}r^2\Omega}{\mathrm{D}t} t = 0. \tag{6.95}$$

With the assumption of rigid rotation, the total angular momentum changes during the evolution, due to local variations in density and radius. It can be imagined that a physical process, not described, allows the global conservation of angular momentum, in which case the *rigid* angular velocity varies as the evolution progresses[12]. Given the assumption of sphericity, the total angular momentum at time t is:

$$\mathcal{M}_\Omega(t) = \frac{2}{3} \int_0^{M_\star(t)} r(t,m)^2 \Omega(t,m) \mathrm{d}m, \tag{6.96}$$

[12]The possibility of global conservation of angular momentum has been implemented in Cesam2k20 at the initiative of M.J.Goupil.

where $\Omega(t,m)$ is the local angular velocity at time t for the Lagrangian abscissa m. With the assumption of ***rigid rotation*** i.e., $\Omega(t,m) \equiv \Omega(t)$ and global conservation of angular momentum i.e., $\mathcal{M}_\Omega(t) \equiv \mathcal{M}_\Omega = cte.$, the angular velocity $\Omega(t+dt)$ at time $t+dt$ satisfies:

$$\mathcal{M}_\Omega(t+dt) = \frac{2}{3}\Omega(t+dt)\int_0^{M_\star(t+dt)} r(t+dt,m)^2 \mathrm{d}m, \tag{6.97}$$

thus:

$$\mathcal{M}_\Omega(t+dt) = \int_0^{M_\star(t)} r(t,m)^2\Omega(t,m)\mathrm{d}m \quad \Big/ \quad \int_0^{M_\star(t+dt)} r(t+dt,m)^2\mathrm{d}m. \tag{6.98}$$

With the assumption of *local conservation of angular momentum*, it is assumed that the CZ, with their extensions by overshoot, are in solid rotation. In these zones, the angular velocity is given by relationships analogous to the previous one in which the integration limits correspond to the boundaries of the CZ. In the radiative parts, the local conservation relation:

$$0 = \frac{\mathrm{D}r^2\Omega}{\mathrm{D}t} = 2r\Omega\frac{\mathrm{D}r}{\mathrm{D}t} + r^2\frac{\mathrm{D}\Omega}{\mathrm{D}t}, \tag{6.99}$$

is discretized as:

$$0 = 2r\Omega\frac{r-r^{(t)}}{\Delta t} + r^2\frac{\Omega-\Omega^{(t)}}{\Delta t} \Longrightarrow \Omega = \frac{r\Omega^{(t)}}{3r-2r^{(t)}}. \tag{6.100}$$

In the absence of a convective core, this relation is indeterminate at the center. It is assumed that the angular velocity at $R=0$ is equal to that determined at the nearest grid point.

The specific rotational kinetic energy is:

$$\epsilon_\Omega = \frac{1}{2}\frac{\mathrm{d}I}{\mathrm{d}m}\Omega^2 = \frac{1}{3}r^2\Omega^2. \tag{6.101}$$

During the evolution, the local angular velocity varies due to the loss of angular momentum, or local or global conservation of angular momentum; these variations lead to those of the airspeed which result in variations of the local specific kinetic energy ϵ_Ω:

$$\frac{\partial\epsilon_\Omega}{\partial t} = \frac{1}{3}\frac{\partial(r\Omega)^2}{\partial t} \simeq \frac{1}{3}\frac{r^2(t+\Delta t)\Omega^2(t+\Delta t) - r(t)^2\Omega(t)^2}{\Delta t} \tag{6.102}$$

It is assumed that these energy variations are, depending on their sign, taken or returned to the medium in the form of heat sources. The energy equation, (6.14), estimated in the `static_m` or `static_r` routine, see §7.76 (Page 226), then becomes:

$$\frac{\partial L}{\partial M} = \epsilon - \frac{\partial U}{\partial t} + \frac{P}{\rho^2}\frac{\partial\rho}{\partial t} - \frac{\partial\epsilon_\Omega}{\partial t}. \tag{6.103}$$

With a solid rotation of nonzero angular velocity, the angular momentum is re-estimated for each intermediate model in the `resout` routine (see Sect. 7.67).

6.6.1 Time evolution with diffusion

Finite element integration

Due to convective mixing, the equations for the diffusion of chemical elements and angular momentum form a fourth-order integro-differential boundary value problem with initial conditions. Although these equations are coupled, Cesam2k20 solves them separately in an alternating manner (method of lines):

1. The diffusion of angular momentum, a fourth-order problem.
2. The diffusion of chemical elements, a parabolic differential problem.

Cesam2k20 uses the finite element method for their numerical integration. Formally, these equations take the form:

$$a\frac{\mathrm{d}y}{\mathrm{d}t} + bG = \frac{\mathrm{d}F}{\mathrm{d}x}, \tag{6.104}$$

where $F(x, y)$, $a(x, y)$, and $b(x, y)$ are functions of the abscissa x and the unknown function $y(x)$.

As previously mentioned, the unknown functions are projected onto a finite-dimensional functional basis. The projections are obtained by writing that the expansions satisfy the differential equations and the boundary conditions. For these basis functions, Cesam2k20 uses piecewise polynomials of order m (degree-$m-1$ polynomials), connecting with a C^i continuity (continuity of derivatives up to order $m-1$). These piecewise polynomials are expressed in the form of normalized polynomial B-splines of order m and dimension d: $\{S_j^m\}_{j=1}^d$. Thus, each of the functions is approximated by an interpolation function of the form:

$$f(x) \simeq \sum_{j=1}^{d} f_j S_j^m(x), \quad \text{so that} \quad \frac{\mathrm{d}f}{\mathrm{d}x} \simeq \sum_{j=1}^{d} f_j \frac{\mathrm{d}S_j^m}{\mathrm{d}x}. \tag{6.105}$$

The knowledge of f_j determines the interpolation function of f with an accuracy that can be estimated (Schumaker, 1981). Stable and accurate algorithms are used to compute the $S_j^m(x)$ and all their derivatives, which are the subject of the `bvald` routine in Cesam2k20, see 8.2.8.

The principle of solving the diffusion equations consists of imposing orthogonality of the basis to its image under the differential operator, i.e., orthogonality to the residual. This is the "Galerkin method". Its application to the cases of chemical species diffusion and angular momentum diffusion is "semi-spectral", as the basis functions do not satisfy the boundary conditions. The numerical solutions obtained are called *weak* solutions. By forming the inner products:

$$\langle f \bullet g \rangle = \int_{x_0}^{x_1} f g \mathrm{d}x, \tag{6.106}$$

we obtain:

$$\left\langle \left(a\frac{\mathrm{d}y}{\mathrm{d}t} + bG \right) \bullet S_j^m \right\rangle = \left\langle \frac{\mathrm{d}F}{\mathrm{d}x} \bullet S_j^m \right\rangle \tag{6.107}$$

By integrating the second inner product by parts, we obtain:

$$\left\langle \frac{\mathrm{d}F}{\mathrm{d}x} \bullet S_j^m \right\rangle = \left[F S_j^m \right]_{x_0}^{x_1} - \left\langle F \bullet \frac{\mathrm{d}S_j^m}{\mathrm{d}x} \right\rangle$$

We have:

$$[F]_{x_0}^{x_1} \equiv \left[F S_j^m \right]_{x_0}^{x_1} = \begin{cases} -F(x_0) & \text{if } j = 1 \\ F(x_1) & \text{if } j = d \\ 0 & \text{otherwise} \end{cases} \tag{6.108}$$

since at the ends of the integration interval, only one of the B-splines is non-zero and equal to one. Ultimately, we obtain the following discretization:

$$\left\langle \left(a\frac{y^{t+1} - y^t}{\Delta t} + bG \right) \bullet S_j^m \right\rangle + \left\langle F \bullet \frac{\mathrm{d}S_j^m}{\mathrm{d}x} \right\rangle - [F]_{x_0}^{x_1} = 0 \tag{6.109}$$

We are reduced - possibly after linearization - to solving a linear system; the B-splines being of bounded support, the linear system is band-diagonal. It is solved by Gauss elimination (partial pivoting) using the gauss_band routine, see Sect. 8.1. The inner products are approximated by numerical integration of Gauss type, following an algorithm inspired by algorithm 5.22, Schumaker (1981, p. 203).

The "integral" nature of the equations results from convective mixing. This is performed in Cesam2k20 through turbulent diffusion with a coefficient $D_{\rm M} \gg 1$.

These provisions have the following advantages:

- The order of the differential equations to be solved is reduced by one.
- Discontinuities of the first derivatives are implicitly taken into account by the integral formulation, (6.109), of the inner products.
- With the use of the B-spline basis, the algorithms remain simple even when boundary shifts occur.

B-spline bases for Petrov-Galerkin

A schematic representation of the nodal vector for $m = 4$, with discontinuity of the first derivative at the arrow point $\Downarrow$, (Schumaker, 1981), is as follows:

$$\begin{array}{cccccccccc} & & & & & \Downarrow & & & & \\ \times & \times & \times & \dots & \times & \times & \times & \dots & \times & \times \\ \times & & & & & \times & & & & \times \\ \times & & & & & \times & & & & \times \\ \times & & & & & & & & & \times \end{array} \tag{6.110}$$

With continuity of the function and
its first two derivatives at all grid points:

$$\begin{array}{cccccccccc} \times & \times & \times & \dots & \times & \times & \times & \dots & \times & \times \\ \times & & & & & & & & & \times \\ \times & & & & & & & & & \times \\ \times & & & & & & & & & \times \end{array} \tag{6.111}$$

6.6.2 Présence de discontinuités

Dans le cas où la fonction à intégrer présente des discontinuités, la base de B-splines sur laquelle elle est projetée, doit inclure ces discontinuités. Avec une discontinuité au point x_d de $]x_0, x_1[$ l'intégrale du produit scalaire, *cf.* équation **??** (Page **??**), est définie par:

$$< f \bullet g >= \int_{x_0}^{x_{d^-}} f g \, dx + \int_{x_{d^-}}^{x_1} f g \, dx, \tag{6.112}$$

Pour le problème différentiel, *cf.* équation **??** (Page **??**), avec le produit scalaire ainsi défini, l'intégration par parties est justifiée, car elle ne porte que sur des fonctions continues. *In fine* pour les B-splines dont le support est contenu dans l'intervalle $[x_0, x_{d^-}[$ on obtient:

$$< \left(a \frac{y^{t+1} - y^t}{\Delta t} + bG \right) \bullet S_j^m > + < F \bullet \frac{\partial S_j^m}{\partial x} > - [F]_{x_0}^{x_{d^-}} = 0, \tag{6.113}$$

et pour celles dont le support est contenu dans l'intervalle $[x_1, x_{d^+}[$:

$$< \left(a \frac{y^{t+1} - y^t}{\Delta t} + bG \right) \bullet S_j^m > + < F \bullet \frac{\partial S_j^m}{\partial x} > - [F]_{x_{d^+}}^{x_1} = 0.$$

Les produits scalaires étant définis par l'équation 6.112 (Page 83).

En un point de discontinuité le vecteur nodal présentera m abscisses identiques. Pour un ordre de B-spline $m = 4$, une représentation schématique du vecteur nodal avec discontinuité au point fléché est:

$$\begin{array}{cccccccccc} & & & & & \Downarrow & & & & \\ \times & \times & \times & \dots & \times & \times & \times & \dots & \times & \times \\ \times & & & & & \times & & & & \times \\ \times & & & & & \times & & & & \times \\ \times & & & & & \times & & & & \times \end{array} \tag{6.114}$$

Cette disposition est équivalente à autant de domaines d'intégration distincts qu'il y a de discontinuités. Cette représentation "regroupée" n'est utile que si les discontinuités apparaissent, disparaissente ou encore se déplacent. Certains systèmes différentiels peuvent présenter simultanément, en divers points du domaine d'intégration des variables continues, des variables continues non dérivables et des variables discontinues. Un exemple est celui du système des équations de diffusion du moment cinétique, *cf.* §6.8 (Page 98). Une représentation schématique d'un vecteur nodal avec une discontinuité de la dérivée première (↓) et une discontinuité (⇓) est:

$$\begin{array}{cccccccccc} & & \downarrow & & & & \Downarrow & & & \\ \times & \times & \times & \times & \dots & \otimes & \times & \times & \dots & \times & \times \\ \times & & \times & & & & \times & & & & \times \\ \times & & \times & & & & \times & & & & \times \\ \times & & & & & & \otimes & & & & \times \end{array} \tag{6.115}$$

Quand le problème différentiel présente, en des abscisses identiques, des variables continues et des variables discontinues[13], Cesam2k20 assure la continuité en égalant les projections des variables concernées sur chacune des deux B-splines localisées de part et d'autre la discontinuité. Dans la représentation schématique précédente, ces B-splines sont identifiées avec le symbole $\otimes$. Pour obtenir autant d'équations que d'inconnues la relation résultant du produit scalaire avec une des B-splines concerné est supprimé. Suppression qui élimine la contribution de cette B-spline à la quantité intégrée, *cf.* §?? (Page ??).

6.7 Diffusion des éléments chimiques

L'équation d'évolution temporelle des espèces chimiques avec diffusion:

$$\frac{Dx_i}{Dt} = \frac{\partial F_i}{\partial m} + \aleph_i, \ 1 \leq i \leq n_{\rm elem} \tag{6.116}$$

contient une partie diffusive, $\frac{\partial F_i}{\partial m}$, et une partie nucléaire, $\aleph_i$. Cette dernière est calculée par la routine de type nuc utilisée, *cf.* §7.6.1 (Page 149). Le flux de particules d'abondance en nombre x_i a pour expression:

$$F_i = 4\pi R^2 \rho \left(4\pi R^2 \rho D_i * \nabla_{\rm m} \mathcal{X} + v_i x_i \right), \tag{6.117}$$

les composantes $d_{i,j}, \ i \neq j$, du vecteur:

$$D_i(m,t) \equiv (d_{i,1}, \dots, d_{i,n_{\rm elem}})^T, \tag{6.118}$$

sont les coefficients, $d_{i,j} = d^\star_{i,j}$, de diffusion microscopique de l'élément d'indice i, par rapport à l'élément d'indice j; la (i,i)-ième composante:

$$d_{i,i} \equiv d^\star_{i,i} + d_{\rm T} + d_{\rm M} \tag{6.119}$$

[13] Tel sera le cas pour le système des équations de diffusion du moment cinétique.

inclut les coefficients de diffusion turbulente $d_{\rm T}$ et de mélange $d_{\rm M}$; $v_i(m,t)$ est la vitesse de sédimentation; on a noté à l'aide du symbole "$*$" le produit scalaire:

$$D_i * \nabla_{\rm m}\mathcal{X} \equiv \sum_j d_{i,j}\frac{\partial x_j}{\partial m}. \tag{6.120}$$

En utilisant la variable spatiale $\nu \equiv (m/M_\odot)^{2/3}$, l'équation 6.116 (Page 84) devient:

$$\frac{\partial x_i}{\partial t} = -\frac{\partial F_i}{\partial m} + \aleph_i = -\frac{2}{3M_\odot\sqrt{\nu}}\frac{\partial F_i}{\partial \nu} + \Psi_i,\ 1 \le i \le n_{\rm elem}. \tag{6.121}$$

Des zones mélangées, i.e. zones convectives éventuellement overshootées, apparaissent, se déplacent, disparaissent au cours de l'évolution. On admet que le mélange convectif est suffisamment efficace pour uniformiser la composition chimique, ainsi:

$$\frac{\partial x_i}{\partial R} = 0. \tag{6.122}$$

Ces conditions entraînent la discontinuité des gradients des abondances aux limites entre zones radiatives et zones mélangées; d'où la nécessité d'introduire des limites internes mobiles et en nombre variable, ce qui complique l'intégration numérique. Grâce à un choix *ad hoc* des bases de B-splines utilisées pour la diffusion des éléments chimiques, les discontinuités des gradients sont implicitement prises en compte.

La présence du mélange convectif change la nature du problème, de différentiel il devient ***intégro-différentiel***. L'équation de transport des éléments chimiques étant du type advection-diffusion, on contourne l'aspect intégral en uniformisant la composition chimique par une diffusion. Pour ce faire, on impose dans les zones mélangées un coefficient de diffusion dont l'ordre de grandeur, grand devant l'unité, correspond au temps de retournement des éléments convectifs. Cesam2k20 utilise $d_{\rm conv} = 10^{13}\ {\rm cm^2 s^{-1}}$. A chaque limite entre une zone mélangée et la zone non mélangée adjacente les coefficients de diffusion sont discontinus avec pour conséquence la discontinuité des gradients des abondances des espèces chimiques.

On représente l'abondance de chaque espèce par un polynôme par morceaux d'ordre $m_c \ge 2$ ***avec discontinuité de la dérivée première*** à chaque limite RZ/CZ, *cf.* § 6.2.1 (Page 59). Pour le calcul numérique, on projette ces polynômes sur une base de B-splines avec discontinuité de la dérivée première: $\mathcal{N} \equiv \{N_k^{m_c}\}_{k=1,K_{\rm X}}$, $\mathcal{N} \subset \mathcal{C}^0[0,\nu_b]$; $K_{\rm X}$ étant la dimension de la base. On a alors:

$$x_i(\nu,t) = \sum_{k=1}^{K_{\rm X}} x_{i,k}(t) N_j^{m_c}(\nu). \tag{6.123}$$

On recherche les *solutions faibles* de l'équation 6.121 (Page 85) à l'aide du formalisme de Galerkin. Les produits scalaires s'écrivent:

$$< \frac{3}{2}M_\odot\sqrt{\nu}\left(\frac{\partial x_i}{\partial t} - \Psi_i\right) \bullet S_j^{m_c} > + < \frac{\partial F_i}{\partial \nu} \bullet S_j^{m_c} >= 0, \tag{6.124}$$

En tenant compte de la continuité des flux F_i, l'intégration par parties conduit à:

$$< \frac{\partial F_i}{\partial \nu} \bullet S_j^{m_c} >= \left[F_i S_j^{m_c}\right]_0^{\nu_{\rm b}} - < F_i \bullet \frac{\partial S_j^{m_c}}{\partial \nu} >, \tag{6.125}$$

avec la condition limite, $\nu \equiv 0$, $F_i(\nu = 0) = 0$ au centre et, à la limite externe de l'enveloppe $\nu = \nu_{\rm b}$. On montre au § 6.7.1 (Page 86):

$$F_i(\nu = \nu_{\rm b}) = \dot{M}(x_{iv} - x_i) \tag{6.126}$$

où x_{iv} est l'abondance par mole de l'espèce i dans le vent stellaire. Par convention, le taux de perte de masse $\dot{M}$ est négatif s'il y a perte, et positif s'il y a gain. Comme à la limite de l'enveloppe, seule la dernière spline $S^{m_c}_{K_X}$, est non nulle

$$\left[F_i S_j^{m_c}\right]_0^{\nu_b} = \dot{M}(x_{iv} - x_i)(\nu_b). \tag{6.127}$$

Physiquement, le vent stellaire ne peut concerner que la limite externe, à défaut de le décrire en détail, on suppose qu'il concerne toute la zone convective externe de masse $M_{\rm ZC}$. Pour simplifier l'algorithme on transpose la partie intégrée dans le terme nucléaire Ψ_i ainsi qu'il est décrit au § 6.7.1 (Page 86).

La continuité de x_i et de F_i permet d'écrire l'équation 6.124 (Page 85) sous la forme utilisée dans Cesam2k20:

$$\begin{aligned} &< \frac{3}{2}\sqrt{\nu}\left(x_i - x_i^{(t)} - \Psi_i \Delta t\right) \bullet S_j^{m_c} > -\frac{\dot{M}}{M_{\rm ZC}}(x_{iv} - x_i)\Delta t + \\ + \quad &< \left[\frac{32\pi^2 R^4 \rho^2\, \Delta t}{3 M_\odot \sqrt{\nu}}\left\{D_i * \nabla_\nu \mathcal{X}\right\} - \frac{4\pi R^2 \rho}{M_\odot}\Delta t\, V_i x_i\right] \bullet \frac{\partial S_j^{m_c}}{\partial \nu} >= 0,\ j = 1, \ldots, K_X, \end{aligned} \tag{6.128}$$

les quantités étant prises à l'instant $t+\Delta t$ sauf $x_i^{(t)}$ pris au temps t. Ce schéma complètement implicite est d'ordre m_c en espace, et du premier ordre en temps.

Au centre, par symétrie, les dérivées spatiales des abondances sont nulles ainsi que leurs vitesses d'advection. Il en est de même pour la limite externe toujours située dans une zone de mélange. Le terme intégré qui est nul a été omis dans l'écriture des équations.

6.7.1 Condition limite externe

On désignera par "vent", indistinctement un apport ou un retrait de masse. A défaut de décrire le processus de création du vent, on suppose que celui-ci est issu de la zone convective externe qui existe toujours. Pendant l'intervalle de temps Δt, l'apport en masse d'élément chimique i à la zone convective $\dot{M} X_{iv} \Delta t$, X_{iv} étant l'abondance par masse de l'élément i au temps t dans le vent.

A l'issue du pas temporel, la fraction de masse de l'élément i dans la zone convective sera:

$$X_i^{t+\Delta t} = \frac{M_{\rm ZC} X_i^t + \dot{M} X_{iv} \Delta t}{\sum_i M_{\rm ZC} X_i^t + \dot{M} X_{iv} \Delta t}, \tag{6.129}$$

où X_i^t et X_{iv} sont respectivement l'abondance par masse de l'élément i au temps t et dans le vent. Les fractions de masse vérifiant $\sum_i X_i^t \equiv 1$ et $\sum_i X_{iv}^t \equiv 1$:

$$\begin{aligned} &X_i^{t+\Delta t} = \frac{M_{\rm ZC} X_i^t + \dot{M} X_{iv} \Delta t}{M_{\rm ZC} + \dot{M}\Delta t} = \frac{X_i^t + \frac{\dot{M}}{M_{\rm ZC}} X_{iv} \Delta t}{1 + \frac{\dot{M}}{M_{\rm ZC}}\Delta t} \sim \\ &X_i^t \left(1 - \frac{\dot{M}}{M_{\rm ZC}}\Delta t\right) + \frac{\dot{M}}{M_{\rm ZC}} X_{iv} \Delta t \left(1 - \frac{\dot{M}}{M_{\rm ZC}}\Delta t\right) = \\ &X_i^t - \frac{\dot{M}}{M_{\rm ZC}} X_i^t \Delta t + \frac{\dot{M}}{M_{\rm ZC}} X_{iv} \Delta t - \left(\frac{\dot{M}}{M_{\rm ZC}} X_{iv} \Delta t\right)^2 \sim X_i^t + \frac{\dot{M}}{M_{\rm ZC}}\left(X_{iv} - X_i^t\right)\Delta t \end{aligned}$$

On obtient pour la dérivée temporelle de l'abondance de l'espèce chimique i:

$$\dot{X}_i = \lim_{\Delta t \to 0} \frac{X_i^{t+\Delta t} - X_i^t}{\Delta t} = \frac{\dot{M}}{M_{\rm ZC}}\left(X_{iv} - X_i^t\right) \iff \dot{x}_i = \frac{\dot{M}}{M_{\rm ZC}}\left(x_{iv} - x_i^t\right), \tag{6.130}$$

où x_{iv} est l'abondance par mole de l'élément i dans le vent. Dans la zone convective externe la variation temporelle de l'isotope X_i est donné par:

$$\dot{x}_i M_{\rm ZC} = \dot{M}(x_{iv} - x_i^t). \tag{6.131}$$

Les abondances ne varient que si la composition chimique du vent est différente de celle des couches externes. Dans la zone convective externe les variations de composition chimique résultant de l'équation 6.131 sont appliquées en addition de celles résultant des réactions thermonucléaires.

Ces dispositions sont appliquées même si la diffusion microscopique des éléments chimique est ignorée.

6.7.2 Chutes de planètoïdes

De façon similaire à ce qui précède, au cours d'une évolution Cesam2k20 permet de simuler une chute de planètoïdes sur un intervalle de temps limité. La composition chimique de la zone convective externe se trouve alors modifiée par des éléments chimiques dont l'abondance et la nature peuvent, éventuellement, différer de celles des conditions initiales et/ou locales. Dans la zone convective externe la variation temporelle de l'isotope X_i est donné par:

$$\dot{x}_i(t) = \frac{N_{\rm P} M_\oplus}{M_{\rm ZC}} \mathcal{P}(t)(x_{iv} - x_i^t). \tag{6.132}$$

$N_{\rm P} > 0$ est le nombre total de planètoïdes de masse terrestre $M_\oplus$ reçus par l'étoile, $\mathcal{P}(t)$ la fonction décrivant la dépendance temporelle, *cf.* §3.15.5 (Page 33).

Cesam2k20 peut tenir compte de l'apport de moment cinétique à la zone convectiveexterne résultant des chutes de planètoïdes, *cf.* §6.8.10 (Page 116).

6.7.3 Formalisme de Burgers

Le formalisme de Burgers (1969), permet de déterminer les coefficients de diffusion microscopique des éléments chimiques. On reformule le système des équations de Burgers pour les variables utilisées par Cesam2k20, les abondances par mole x_i et le poids moléculaire moyen μ. Le principe de la méthode numérique de Burgers, **?**, consiste à remarquer que la résolution d'un système linéaire permet d'écrire la vitesse de diffusion w_i de la particule i sous la forme:

$$w_i = v_i + \sum_j b_{ij} \frac{{\rm d}x_j}{{\rm d}x}. \tag{6.133}$$

? n'ont pas tenu compte des "residual heat flow vector" qui furent introduits par la suite par **?** et repris par **?**.

On utilise les notations suivantes:

- c: nombre d'ions, il y a $c+1$ espèces de particules, i.e. ions+ électrons,
- E: champ électrique,
- e: charge de l'électron, en unité électrostatique, et indice de la particule "électron",
- $g_{\rm G}$: accélération due à la gravité,
- g_Ω: accélération centrifuge,
- $g_{\rm e}$: accélération effective,
- $g_{\rm Ri}$: accélération radiative sur la particule i,
- g_i: accélération due à la gravité effective et aux forces radiatives sur i,
- $g_{\rm t}$: accélération totale,
- $g_{j,i}$: poids statistique du fondamental du niveau d'ionisation j de l'élément chimique i,

- $m_e = \nu_e m_u$: masse de l'électron,
- $m_i \equiv \nu_i m_u$: masse de l'élément chimique i,
- $m_{ij} \equiv \frac{m_i m_j}{m_i + m_j}$: masse réduite des particules d'indices i et j,
- $m_u = 1/N_0$: masse atomique unité, inverse du nombre d'Avogadro,
- n_e: nombre d'électrons par unité de volume,
- n_i: nombre d'ions de l'élément chimique $i \neq e$ par unité de volume,
- $n_{j,i}$: nombre d'ions de l'élément chimique i par unité de volume, dans l'état d'ionisation $j = 0, \ldots, Z_i$,
- r: rayon, variable d'espace eulérienne,
- $T,\ P,\ \rho$: température, pression gaz parfait, densité,
- t: temps,
- $\mathcal{V} = \frac{\partial r}{\partial t}$: vitesse d'entraînement du fluide, $\mathcal{V} \sim 0$ (hypothèse de l'équilibre quasi-statique),
- w_i: vitesse de diffusion de la particule i, par rapport à la vitesse moyenne du fluide,
- X_i: proportion en masse de l'abondance de l'élément chimique i, $\sum_i X_i \equiv 1$,
- $x_i \equiv X_i/\nu_i$: proportion par mole de l'abondance de l'élément chimique i,
- $x_{j,i}$: taux d'ionisation du niveau d'ionisation j de l'ion i,
- Z_i charge de la particule i, $Z_e = -1$,
- $\overline{Z_i}$: charge moyenne de l'élément i avec ionisation,
- η: paramètre de dégénérescence,
- μ: poids moléculaire moyen,
- ν_e: "masse atomique" de l'électron, $\nu_e = m_e/m_u$,
- ν_i: masse atomique de l'élément chimique i,
- $\chi_{j,i}$: potentiel d'ionisation du niveau d'ionisation j de l'élément i.

Problem: *Les notations ne sont pas totalement cohérentes en ce qui concerne les électrons.*

6.7.4 Charge moyenne des ions

On utilise l'équation de Saha (Cox & Giuli, 1968, eq. 15-30) dans laquelle on limite les fonctions de partition aux poids statistiques des niveaux fondamentaux:

$$\frac{n_{j-1,i}}{n_{j,i}} = \frac{g_{j-1,i}}{g_{j,i}} \exp\left(\eta - \Delta\mu + \frac{\chi_{j,i}}{kT}\right) \equiv \varphi_{j,i},\ j = 1, \ldots, Z_i. \tag{6.134}$$

Le taux de dégénérescence η vérifie:

$$F_{\frac{1}{2}}(\eta) = \frac{n_e}{4\pi}\left(\frac{h^2}{2m_e kT}\right)^{\frac{3}{2}}, \tag{6.135}$$

$F_{\frac{1}{2}}(\eta)$ est la fonction de Fermi-Dirac (Clayton, 1968, eq. 2-57). $\Delta\mu$ est une correction numérique introduite par Eggleton et al. (1973), pour simuler l'ionisation de pression, et éviter la recombinaison à haute température (**?**, Sect. 14.6):

$$\Delta\mu = n_e c_F \left(\frac{a_0}{\bar{Z}}\right)^3 (1 + \frac{20\chi_0}{kT}),\ a_0 = \frac{h^2}{4\pi^2 m_e e^2},\ \bar{Z} = X + 2Y + 8Z,\ c_F = 15,\ \chi_0 = 13.6. \quad (6.136)$$

Problem: *μ n'est pas, dans ce paragraphe, le poids moléculaire moyen.*

L'expérience des calculs a montré que le formalisme d'Eggleton Eggleton et al. (1973), même avec la modification de **?**, eq. 4, donnait des taux d'ionisation moyens avec des paliers et des gradients beaucoup plus marqués que ceux obtenus avec une meilleure prise en compte de l'ionisation de pression. Ainsi, au centre du Soleil, le formalisme de Eggleton et al. (1973) donne un taux d'ionisation de 100% pour Fe XXV, alors que le calcul plus précis de Gabriel (1997) ne donne que 85%.

Pour tenir compte de l'ionisation de pression et éviter la recombinaison, la routine saha de Cesam2k20 utilise une correction numérique dérivée de l'analyse de Clayton (1968, p. 140-145). Le potentiel d'ionisation est réduit lorsque la distance moyenne entre les ions devient de l'ordre de grandeur de la longueur de Debye:

$$R_{\rm D} = \sqrt{\frac{kT}{4\pi e^2 \rho N_0 \zeta}},\quad \zeta = \sum_{i\neq e} \overline{Z_i}(\overline{Z_i} + 1)x_i \quad (6.137)$$

et se réduit[14] à:

$$\chi'_{j,i} = \max(0, \chi_{i,j} - \chi^{\rm C}_{i,j}),\quad \chi^{\rm C}_{i,j} = \frac{je^2}{R_{\rm D}}. \quad (6.138)$$

Cette correction appliquée seulement sur le potentiel d'ionisation s'est avérée insuffisante pour éviter la recombinaison à haute température, en particulier, de l'ion HII. La routine saha utilise un ajustement *numérique* agissant à la fois sur le potentiel d'ionisation et le quotient des fonctions de partition; celles-ci devant être tronquées de façon cohérente avec la diminution des potentiels d'ionisation, *cf.* Clayton (1968, fig. 2-16). Celà n'est pas possible avec les fonctions de partition limitées aux poids statistiques des niveaux fondamentaux. L'artifice consiste en une diminution progressive de 1 à 0 du rapport des poids statistiques des niveaux fondamentaux $g_{j-1,i}/g_{j,i}$, dès que la quantité:

$$x = \frac{\chi_{i,j}}{\chi'_{j,i}} - 1 \quad (6.139)$$

devient inférieure à une valeur p fixée, pour le niveau d'ionisation $j = 1, \ldots, Z_i$ de l'élément chimique i. L'expérience des calculs a montré que la valeur $p = 4$ permettait de retrouver, pour des modèles d'étoiles de la séquence principale de $1M_\odot$ et $1.4M_\odot$ respectivement, des taux d'ionisation voisins de ceux obtenus avec un calcul plus exact, mais beaucoup plus lourd, de la correction de pression d'ionisation. La fonction de raccordement $f(x)$ utilisée est un morceau de cubique sur $[0, p]$, de pente nulle en $x = 0$ et $x = p$:

$$f(x) = 0 \text{ si } x \leq 0,\ \left(\frac{x}{p}\right)^2 \left[-2\left(\frac{x}{p}\right) + 3\right] \text{ si } x \in [0, p],\ f(x) = 1, \text{ si } x \geq p. \quad (6.140)$$

$$f(x) = 0 \text{ si } x \leq 0,\ \left(\frac{x}{p}\right)^2 \left[-2\left(\frac{x}{p}\right) + 3\right] \text{ si } x \in [0, p],\ f(x) = 1, \text{ si } x \geq p.$$

Avec ce formalisme approximatif, l'équation de Saha 6.134 devient:

$$\frac{n_{j-1,i}}{n_{j,i}} = \frac{g_{j-1,i}}{g_{j,i}} f(x) \exp\left(\eta + \frac{\chi_{j,i}}{kT}\right) \equiv \varphi_{j,i},\ j = 1, \ldots, Z_i. \quad (6.141)$$

[14] je est la charge de l'ion.

On obtient ainsi l'ionisation totale du niveau $j-1$ lorsque $g_{j-1,i}/g_{j,i}f(x)=0$ i.e. dès que $\chi_{i,j}\leq\chi^{\rm C}_{i,j}$.

Le nombre de particules $n_i,\ i\neq e$ est:

$$\begin{aligned} n_i &= \sum_{j=0}^{Z_i} n_{j,i} = n_{Z_i,i}\left(1+\sum_{j=0}^{Z_i-1}\frac{n_{j,i}}{n_{Z_i,i}}\right) = n_{Z_i,i}\left(1+\sum_{j=0}^{Z_i-1}\prod_{k=j+1}^{Z_i}\varphi_{k,i}\right) \\ &= n_{Z_i,i}\left(1+\frac{n_{0,i}}{n_{Z_i,i}}+\frac{n_{1,i}}{n_{Z_i,i}}+\ \ldots\ +\frac{n_{Z_i-1,i}}{n_{Z_i,i}}\right) \\ &= n_{Z_i,i}\left(1+\varphi_{1,i}\varphi_{2,i}\varphi_{3,i}\ \ldots\ \varphi_{Z_i,i}+\varphi_{2,i}\varphi_{3,i}\ \ldots\ \varphi_{Z_i,i}+\ \ldots\ +\varphi_{Z_i,i}\right) \\ &= n_{Z_i,i}\left(1+\varphi_{Z_i,i}\left(1+\varphi_{Z_i-1,i}\left(1+\varphi_{Z_i-2,i}\left(1+\ldots\varphi_{1,i}\right)\right)\right)\right) \end{aligned}$$

et comme:

$$\varphi_{j+1,i}=\frac{n_{j,i}}{n_{j+1,i}}=\frac{n_{j,i}}{n_i}\frac{n_i}{n_{j+1,i}}=\frac{x_{j,i}}{x_{j+1,i}}, \tag{6.142}$$

on a:

$$\begin{aligned} x_{Z_i,i}\equiv\frac{n_{Z_i,i}}{n_i} &= \frac{1}{\left(1+\varphi_{Z_i,i}\left(1+\varphi_{Z_i-1,i}\left(1+\varphi_{Z_i-2,i}\left(1+\ldots\varphi_{1,i}\right)\right)\right)\right)}, \\ x_{j,i} &= \varphi_{j+1,i}\,x_{j+1,i},\ j=Z_i-1,\ldots,0. \end{aligned}$$

La charge moyenne de l'élément chimique i et le nombre d'électrons par mole et par unité de volume s'écrivent respectivement:

$$\overline{Z_i}\equiv\sum_{j=1}^{Z_i} j x_{j,i},\ x_e=\sum_{j\neq e}\overline{Z_i}x_i,\ n_e=\rho N_0 x_e=\rho N_0\sum_{j\neq e}\overline{Z_i}x_i. \tag{6.143}$$

On a aussi:

$$X_i=\frac{n_i\nu_i m_u}{\sum_j n_j\nu_j m_u}=\frac{n_i\nu_i}{\rho N_0},\ x_i=\frac{X_i}{\nu_i}=\frac{n_i}{\rho N_0}. \tag{6.144}$$

6.7.5 Equation de diffusion des espèces chimiques

Pour la particule i, l'équation de diffusion s'écrit (**?**):

$$\frac{\partial n_i}{\partial t}=\frac{1}{r^2}\frac{\partial}{\partial r}(r^2 n_i w_i)+\left(\frac{\partial n_i}{\partial t}\right)_{\rm nucl.} \tag{6.145}$$

et les équations de Burgers (1969, eq. 18-4,5) , voir aussi **??**:

$$\frac{{\rm d}P_i}{{\rm d}r}-\frac{\rho_i}{\rho}\frac{{\rm d}P}{{\rm d}r} - n_i\overline{Z_i}eE=\sum_j K_{ij}(w_j-w_i)+\sum_j K_{ij}z_{ij}\frac{m_j r_i-m_i r_j}{m_i+m_j} \tag{6.146}$$

$$\begin{aligned} \frac{5}{2}n_i k\frac{{\rm d}T}{{\rm d}r} = &-\frac{5}{2}\sum_j K_{ij}z_{ij}\frac{m_j(w_j-w_i)}{m_i+m_j}-\frac{2}{5}K_{ii}z''_{ii}r_i \\ &-r_i\sum_{j\neq i}K_{ij}\frac{3m_i^2+m_j^2 z'_{ij}+\frac{4}{5}m_i m_j z''_{ij}}{(m_i+m_j)^2} \\ &+\sum_{j\neq i}K_{ij}\frac{m_i m_j(3+z'_{ij}-\frac{4}{5}z''_{ij})}{(m_i+m_j)^2}r_j. \end{aligned} \tag{6.147}$$

Les équations de conservation dynamique et statique de la masse et de la charge ont respectivement pour expression:

$$\sum_i\nu_i n_i w_i=0,\ e\sum_i\overline{Z_i}n_i w_i=0,\ \sum_i\overline{Z_i}n_i=0,\ \sum_i X_i=\sum_i\nu_i x_i=1. \tag{6.148}$$

La densité partielle ρ_i de la particule i est définie par:

$$\rho_i = \rho X_i = \rho x_i \nu_i = n_i \nu_i m_u,\ \sum_i \rho_i = \rho = \sum_i n_i \nu_i m_u, \tag{6.149}$$

et:

$$n_i = \rho N_0 \frac{X_i}{\nu_i} = \rho N_0 x_i. \tag{6.150}$$

Le second terme du membre de gauche de l'équation (6.146) correspond à la force externe qui s'exerce sur la particule i, à savoir, la gravité corrigée de l'accélération radiative.

6.7.6 Intégrales de collision

Les coefficients de résistance du flux de chaleur $z_{ij}, z'_{ij}, z''_{ij}$ sont définis à partir des intégrales de collision $\Omega_{ij}^{(kl)}$, *cf.* Paquette **?** Eq. (23) - (25):

$$z_{ij} = 1 - \frac{2}{5}\frac{\Omega_{ij}^{(12)}}{\Omega_{ij}^{(11)}},\ z'_{ij} = 2.5 - \frac{2}{5}\frac{5\Omega_{ij}^{(12)} - \Omega_{ij}^{(13)}}{\Omega_{ij}^{(11)}},\ z''_{ij} = \frac{\Omega_{ij}^{(22)}}{\Omega_{ij}^{(11)}}, \tag{6.151}$$

qui sont déterminées à partir de tabulations. Paquette **?** ont écrit $\Omega_{ij}^{(kl)}$ sous la forme, Eq.(65):

$$\Omega_{ij}^{(kl)} \equiv F_{ij}^{(kl)} \epsilon_{ij},\ \epsilon_{ij} \equiv \frac{e^4}{4}\sqrt{\frac{\pi}{2k^3}}\ \frac{\overline{Z_i}^2 \overline{Z_j}^2}{\sqrt{m_{ij}}}\ \frac{1}{\sqrt{T^3}}. \tag{6.152}$$

Pour $(kl) = (11),\ (12),\ (13),\ (22)$, et selon qu'il s'agit d'un potentiel attractif, i.e. électron/ion, ou répulsif, i.e. ion/ion ou électron/électron, Paquette **?** ont calculé et tabulé les coefficients d'interpolation par splines naturelles de $\ln F_{ij}^{(kl)}$ en fonction de la variable $\psi_{ij} \equiv \ln\left[\ln\left(1+\gamma_{ij}^2\right)\right]$ avec:

$$\gamma_{ij} \equiv \frac{4kT\lambda}{\overline{Z_i Z_j} e^2},\ \lambda \equiv \max\left\{\left(\frac{kT}{4\pi e^2 \sum_i n_i \overline{Z_i}}\right)^{\frac{1}{2}}, \left(\frac{3}{4\pi \sum_{\text{ions}} n_i}\right)^{\frac{1}{3}}\right\}. \tag{6.153}$$

Les quantités K_{ij} sont les coefficients de résistance, ils représentent les effets des collisions entre les particules de types i et j, ils ont pour expression *cf.* Michaud & Proffitt (1993):

$$K_{ij} = \frac{16}{3} n_i n_j m_{ij} \Omega_{ij}^{(11)}. \tag{6.154}$$

6.7.7 Equations de Burgers pour les x_i et μ.

En remplaçant la charge de chaque ion par sa charge moyenne tenant compte de son taux d'ionisation, le poids moléculaire moyen est:

$$\mu^{-1} = \sum_{j\neq e}(1+\overline{Z_j})\frac{X_j}{\nu_j} = \sum_{j\neq e}(1+\overline{Z_j})x_j. \tag{6.155}$$

Pour les dérivations, on utilisera:

$$d\mu = -\mu^2 \sum_{j\neq e}(1+\overline{Z_j})dx_j,\ \frac{\partial \mu}{\partial x_k} = -\mu^2(1+\overline{Z_k}),\ k \neq e. \tag{6.156}$$

Avec l'hypothèse du gaz parfait, pour chaque espèce i, les pressions totale et partielles s'écrivent:

$$P = \frac{\rho \mathcal{R} T}{\mu} + P_{\text{R}} = \sum_i P_i, \tag{6.157}$$

$$P_i = n_i kT + P_{i\mathrm{R}} = \rho x_i N_0 kT + P_{i\mathrm{R}} = \frac{P\mu}{\mathcal{R}T} x_i N_0 kT + P_{i\mathrm{R}} = P\mu x_i + P_{i\mathrm{R}}, \tag{6.158}$$

puisque la constante des gaz parfaits vérifie $\mathcal{R} \equiv kN_0$. On a donc:

$$n_i k = \frac{P}{T}\mu x_i. \tag{6.159}$$

En posant, pour les électrons:

$$x_e \equiv \frac{n_e}{\rho N_0} = \frac{\sum_{i\neq e} \overline{Z_i} n_i}{\rho N_0} = \sum_{i\neq e} \overline{Z_i} x_i, \ \frac{\partial x_e}{\partial x_k} = \overline{Z_k}, \ k \neq e, \tag{6.160}$$

comme $P_{e\mathrm{R}} = 0$, on retrouve bien la pression électronique:

$$P\mu x_e = \frac{\rho \mathcal{R} T}{\mu}\mu \frac{n_e}{\rho N_0} = n_e kT = P_e. \tag{6.161}$$

L'hypothèse d'équilibre quasi-statique et $\rho_i = \rho \nu_i x_i$ (Eq. 6.149) entrainent:

$$\frac{\rho_i}{\rho}\frac{\mathrm{d}P}{\mathrm{d}r} = x_i \nu_i \rho g_{\mathrm{t}}, \tag{6.162}$$

et:

$$n_i k \frac{\mathrm{d}T}{\mathrm{d}r} = \frac{P}{T}\mu x_i \frac{T}{P}\frac{\partial \ln T}{\partial \ln P}\frac{\mathrm{d}P}{\mathrm{d}r} = \mu x_i \nabla \rho g_{\mathrm{t}}, \tag{6.163}$$

d'où:

$$\begin{aligned}
\frac{\mathrm{d}P_i}{\mathrm{d}r} &= \frac{\mathrm{d}}{\mathrm{d}r} P\mu x_i + \frac{\mathrm{d}P_{i\mathrm{R}}}{\mathrm{d}r} = \mu x_i \frac{\mathrm{d}P}{\mathrm{d}r} + \frac{\mathrm{d}P_{i\mathrm{R}}}{\mathrm{d}r} + \mu P \frac{\mathrm{d}x_i}{\mathrm{d}r} + P x_i \frac{\mathrm{d}\mu}{\mathrm{d}r} \\
&= x_i \rho \mu g_{\mathrm{t}} + \rho_i g_{\mathrm{R}i} + \mu P \frac{\mathrm{d}x_i}{\mathrm{d}r} - P x_i \mu^2 \sum_{j\neq e} (1 + \overline{Z_j}) \frac{\mathrm{d}x_j}{\mathrm{d}r} \\
&= x_i \rho (\mu g_{\mathrm{t}} + \nu_i g_{\mathrm{R}i}) + P\mu \left(\frac{\mathrm{d}x_i}{\mathrm{d}r} - x_i \mu \sum_{j\neq e} (1 + \overline{Z_j}) \frac{\mathrm{d}x_j}{\mathrm{d}r} \right)
\end{aligned}$$

On a par ailleurs:

$$n_i \overline{Z_i} eE = \rho N_0 x_i \overline{Z_i} eE, \ \sum_i \nu_i x_i w_i = 0, \ \sum_i \overline{Z_i} x_i w_i = 0. \tag{6.164}$$

On introduit les relations précédentes dans les équations de Burgers (6.146), (6.148) et de conservation (6.148). Les coefficients de résistance K_{ij} sont de la forme:

$$K_{ij} = \frac{16}{3} n_i n_j m_{ij} \Omega_{ij}^{(11)} = \frac{16}{3} (\rho N_0)^2 x_i x_j m_{ij} \Omega_{ij}^{(11)} = k_{ij} x_i x_j \rho^2, \ k_{ij} \equiv \frac{16 N_0^2}{3} m_{ij} \Omega_{ij}^{(11)}. \tag{6.165}$$

En fonction des x_i et de μ les équations de Burgers s'écrivent:

$$
\begin{aligned}
& x_i\rho\left(\mu g_{\rm t}-\nu_i\left(g_{\rm t}-g_{{\rm R}i}\right)\right)+P\mu\left(\frac{{\rm d}x_i}{{\rm d}r}-x_i\mu\sum_{j\neq e}(1+\overline{Z_j})\frac{{\rm d}x_j}{{\rm d}r}\right)-\rho N_0 x_i\overline{Z_i}eE\\
&=\rho^2\left\{x_i\sum_j k_{ij}x_j(w_j-w_i)-x_i\sum_j k_{ij}x_j z_{ij}\frac{m_i r_j-m_j r_i}{m_i+m_j}\right\},\ i=1,\ldots,c,\\
&-x_i\rho\nabla\mu g_{\rm t}=\rho^2\left\{x_i\sum_j k_{ij}x_j z_{ij}\frac{m_j(w_j-w_i)}{m_i+m_j}+x_i\frac{4}{25}k_{ii}x_i z''_{ii}r_i\right.\\
&+\frac{2}{5}r_i x_i\sum_{j\neq i}k_{ij}x_j\frac{3m_i^2+m_j^2z'_{ij}+\frac{4}{5}m_i m_j z''_{ij}}{(m_i+m_j)^2}\\
&\left.-\frac{2}{5}x_i\sum_{j\neq i}k_{ij}x_j r_j\frac{m_i m_j(3+z'_{ij}-\frac{4}{5}z''_{ij})}{(m_i+m_j)^2}\right\},\ i=1,\ldots,c,e,
\end{aligned}
$$

ou encore, en posant:

$$
\nu_{i{\rm R}}\equiv\nu_i\left(1-\frac{g_{{\rm R}i}}{g_{\rm t}}\right),\ \frac{\partial\nu_{i{\rm R}}}{\partial x_i}=-\frac{\nu_i}{g_{\rm t}}\left(\frac{\partial g_{{\rm R}i}}{\partial x_j}-\frac{g_{{\rm R}i}}{g_{\rm t}}\frac{\partial g_{\rm t}}{\partial x_j}\right) \tag{6.166}
$$

puis arrangements et division de chaque membre par $x_i\rho^2$:

$$
\begin{aligned}
\frac{g_{\rm t}}{\rho}(\mu-\nu_{i{\rm R}}) &+ \frac{\mathcal{R}T}{\rho}\left(\frac{1}{x_i}\frac{{\rm d}x_i}{{\rm d}r}-\mu\sum_{j\neq e}(1+\overline{Z_j})\frac{{\rm d}x_j}{{\rm d}r}\right)=\sum_{j\neq i}k_{ij}x_j(w_j-w_i)\\
&- \sum_{j\neq i}k_{ij}x_j z_{ij}\frac{m_i r_j-m_j r_i}{m_i+m_j}+\frac{\overline{Z_i}eN_0E}{\rho},\ i=1,\ldots,c, \qquad (6.167)\\
-\frac{\mu\nabla g_{\rm t}}{\rho} &= \sum_{j\neq i}k_{ij}x_j z_{ij}\frac{m_j(w_j-w_i)}{m_i+m_j}+\\
&+ r_i\left(\frac{4}{25}k_{ii}x_i z''_{ii}+\frac{2}{5}\sum_{j\neq i}k_{ij}x_j\frac{3m_i^2+m_j^2z'_{ij}+\frac{4}{5}m_i m_j z''_{ij}}{(m_i+m_j)^2}\right)-\\
&- \frac{2}{5}\sum_{j\neq i}k_{ij}x_j r_j\frac{m_i m_j(3+z'_{ij}-\frac{4}{5}z''_{ij})}{(m_i+m_j)^2},\ i=1,\ldots,c,e. \qquad (6.168)
\end{aligned}
$$

Les données sont la température T, la densité ρ, les gravités $g_{\rm t}$ et $g_{{\rm R}i}$, le gradient ∇, les abondances des ions $x_i,\ i=1,\ldots,c$. Des abondances, on déduit le nombre d'électrons et le poids moléculaire moyen.

Problem: *Faute de mieux, le poids moléculaire moyen, les pressions partielles, ne peuvent pas être cohérents avec ceux utilisés dans l'équation d'état retenue par Cesam2k20.*

Les inconnues sont les $c+1$ vitesses de diffusion $w_1,\ldots,w_c,w_e$, les $c+1$ flux de chaleur $r_1,\ldots,r_c,r_e$ et le champ électrique E, soit $2c+3$ inconnues. On ajoute aux $2c+1$ (6.167) et (6.168), les deux équations de conservation:

$$
\sum_{i=1}^{c}\nu_i x_i w_i=0,\ \sum_{i=1}^{e}\overline{Z_i}x_i w_i=0. \tag{6.169}
$$

Les (6.167), (6.168) écrites pour les x_i et μ et (6.169) forment le système linéaire:

$$
A\omega=\gamma+G\,D_x,\ G\equiv\frac{\mathcal{R}T}{\rho}\Delta, \tag{6.170}
$$

on a noté:

$$\omega \equiv (w_1, \ldots, w_c, w_e, r_1, \ldots, r_c, r_e, E)^T, \tag{6.171}$$

$$D_x \equiv (\frac{\mathrm{d}x_1}{\mathrm{d}r}, \ldots, \frac{\mathrm{d}x_c}{\mathrm{d}r}, 0, \ldots, 0)^T, \tag{6.172}$$

$$\gamma \equiv \frac{g_\mathrm{t}}{\rho} \left(\mu - \nu_{1\mathrm{R}}, \ldots, \mu - \nu_{c\mathrm{R}}, -\mu\nabla, \ldots, -\mu\nabla, 0, 0\right)^T. \tag{6.173}$$

$$\frac{\partial}{\partial x_j}\left(\frac{\mu g_\mathrm{t}\nabla}{\rho}\right) = \frac{g_\mathrm{t}\nabla}{\rho}\frac{\partial \mu}{\partial x_j} + \frac{\mu\nabla}{\rho}\frac{\partial g_\mathrm{t}}{\partial x_j} + \left\{\frac{\mu g_\mathrm{t}}{\rho}\frac{\partial \nabla}{\partial x_1} - \frac{\mu g_\mathrm{t}\nabla}{\rho^2}\frac{\partial \rho}{\partial x_1}\right\}_{j=1}, \tag{6.174}$$

$$\frac{\partial}{\partial x_j}\left(\frac{g_\mathrm{t}(\mu - \nu_{i\mathrm{R}})}{\rho}\right) = \frac{(\mu - \nu_{i\mathrm{R}})}{\rho}\frac{\partial g_\mathrm{t}}{\partial x_j} + \frac{g_\mathrm{t}}{\rho}\left(\frac{\partial \mu}{\partial x_j} - \frac{\partial \nu_{i\mathrm{R}}}{\partial x_j}\right) + \left\{\frac{g_\mathrm{t}}{\rho^2}(\mu - \nu_{i\mathrm{R}})\frac{\partial \rho}{\partial x_1}\right\}_{j=1}. \tag{6.175}$$

$$\Delta \equiv \mathrm{Diag}\left(\frac{1}{x_1}, \frac{1}{x_2}, \ldots, \frac{1}{x_c}, 0, \ldots, 0\right) - \mu\mathcal{M}, \tag{6.176}$$

$$\mathcal{M} \equiv \begin{pmatrix}
(1+\bar{Z}_1) & (1+\bar{Z}_2) & (1+\bar{Z}_3) & \ldots & (1+\bar{Z}_c) & 0 & 0 & \ldots & 0 & 0 \\
(1+\bar{Z}_1) & (1+\bar{Z}_2) & (1+\bar{Z}_3) & \ldots & (1+\bar{Z}_c) & 0 & 0 & \ldots & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
(1+\bar{Z}_1) & (1+\bar{Z}_2) & (1+\bar{Z}_3) & \ldots & (1+\bar{Z}_c) & 0 & 0 & \ldots & 0 & 0 \\
(1+\bar{Z}_1) & (1+\bar{Z}_2) & (1+\bar{Z}_3) & \ldots & (1+\bar{Z}_c) & 0 & 0 & \ldots & 0 & 0 \\
0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0
\end{pmatrix}. \tag{6.177}$$

La structure de la matrice A, de dimension $[2c+3 \times 2c+3]$, est la suivante:

$$\begin{pmatrix} A_w & A_r & A_e \\ A_c & 0 & 0 \end{pmatrix}, \tag{6.178}$$

les dimensions des sous matrices sont respectivement $A_w[2c+1\times c+1]$, $A_r[2c+1\times c+1]$, $A_e[2c+1\times 1]$ et $A_c[2\times c+1]$. En notant:

$$p_{ij} \equiv k_{ij}x_j,\ q_{ij} \equiv \frac{k_{ij}x_j z_{ij} m_j}{m_i + m_j} = p_{ij}\frac{z_{ij}m_j}{m_i + m_j}, \tag{6.179}$$

$$A_w \equiv \begin{pmatrix}
-\sum_{j\neq 1} p_{1j} & p_{12} & p_{13} & \ldots & p_{1c} & p_{1e} \\
p_{21} & -\sum_{j\neq 2} p_{2j} & p_{23} & \ldots & p_{2c} & p_{2e} \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
p_{c1} & p_{c2} & p_{c3} & \ldots & -\sum_{j\neq c} p_{cj} & p_{ce} \\
-\sum_{j\neq 1} q_{1j} & q_{12} & q_{13} & \ldots & q_{1c} & q_{1e} \\
q_{21} & -\sum_{j\neq 2} q_{2j} & q_{23} & \ldots & q_{2c} & q_{2e} \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
q_{c1} & q_{c2} & q_{c3} & \ldots & -\sum_{j\neq c} q_{cj} & q_{ce}
\end{pmatrix}, \tag{6.180}$$

avec les notations:

$$\overline{q_{ij}} \equiv -\frac{k_{ij}x_j z_{ij} m_i}{m_i + m_j} = -q_{ij}\frac{m_i}{m_j}, \tag{6.181}$$

$$d_i \equiv \frac{4}{25}k_{ii}x_i z''_{ii} + \frac{2}{5}\sum_{j\neq i} k_{ij}x_j \frac{3m_i^2 + m_j^2 z'_{ij} + \frac{4}{5}m_i m_j z''_{ij}}{(m_i+m_j)^2}, \tag{6.182}$$

$$f_{ij} \equiv -\frac{2}{5}k_{ij}x_j \frac{m_i m_j (3 + z'_{ij} - \frac{4}{5}z''_{ij})}{(m_i+m_j)^2}, \tag{6.183}$$

$$A_r \equiv \begin{pmatrix} \sum_{j\neq 1} q_{1j} & \overline{q_{12}} & \overline{q_{13}} & \dots & \overline{q_{1c}} & \overline{q_{1e}} \\ \overline{q_{21}} & \sum_{j\neq 2} q_{2j} & \overline{q_{23}} & \dots & \overline{q_{2c}} & \overline{q_{2e}} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \overline{q_{c1}} & \overline{q_{c2}} & \overline{q_{c3}} & \dots & \sum_{j\neq c} q_{cj} & \overline{q_{ce}} \\ d_1 & f_{12} & f_{13} & \dots & f_{1c} & f_{1e} \\ f_{21} & d_2 & f_{23} & \dots & f_{2c} & f_{2e} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ f_{c1} & f_{c2} & f_{c3} & \dots & d_c & f_{ce} \\ f_{e1} & f_{e2} & f_{e3} & \dots & f_{ec} & d_e \end{pmatrix}, \tag{6.184}$$

$$A_e \equiv \frac{N_0 e}{\rho}(\bar{Z}_1, \dots, \bar{Z}_c, 0, \dots, 0)^T, \tag{6.185}$$

$$A_c \equiv \begin{pmatrix} \nu_1 x_1 & \nu_2 x_2 & \dots & \nu_c x_c & 0 & 0 & \dots & 0 \\ \bar{Z}_1 x_1 & \bar{Z}_2 x_2 & \dots & \bar{Z}_c x_c & \bar{Z}_e x_e & 0 & \dots & 0 \end{pmatrix}. \tag{6.186}$$

La solution s'écrit:

$$\omega = A^{-1}\gamma + A^{-1}G\, D_x = v + B\, D_x \Rightarrow, \tag{6.187}$$

ou encore, pour les ions, $i = 1, \dots, c$:

$$w_i = v_i + \sum_{j=1}^{c} b_{ij}\frac{\mathrm{d}x_j}{\mathrm{d}r},\ i = 1, \dots, c, \tag{6.188}$$

les v_i sont les c premiers coefficients de:

$$v = A^{-1}\gamma, \tag{6.189}$$

les $b_{ij},\ (i, j = 1, \dots, c)$, sont les coefficients des c premières lignes et colonnes de

$$B = A^{-1}G. \tag{6.190}$$

Pour l'ion i l'équation de diffusion devient:

$$\frac{\partial n_i}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 n_i w_i) + \left(\frac{\partial n_i}{\partial t}\right)_{\text{nucl.}} =$$

$$\frac{\partial \rho N_0 x_i}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left\{N_0 r^2 \rho x_i \left(\sum_{j=1}^{c} b_{ij}\frac{\partial x_j}{\partial r} + v_i\right)\right\} + \left(\frac{\partial \rho N_0 x_i}{\partial t}\right)_{\text{nucl.}}$$

ou encore, en notant $\frac{D\star}{Dt} \equiv \frac{\partial\star}{\partial t} + \mathcal{V} * \nabla\star$ l'opérateur de dérivation lagrangien "en suivant le mouvement":

$$\frac{Dx_i}{Dt} = \frac{4\pi}{4\pi r^2\rho}\frac{\partial}{\partial r}\left\{\frac{4\pi r^4\rho^2}{4\pi r^2\rho}x_i\left(\sum_{j=1}^{c} b_{ij}\frac{\partial x_j}{\partial r} + v_i\right)\right\} + \left(\frac{Dx_i}{Dt}\right)_{\text{nucl.}} \Longrightarrow \tag{6.191}$$

$$\frac{Dx_i}{Dt} = \frac{4\pi}{4\pi r^2\rho}\frac{\partial}{\partial r}\left\{\frac{4\pi r^4\rho^2}{4\pi r^2\rho}x_i\left(\sum_{j=1}^{c} b_{ij}\frac{\partial x_j}{\partial r} + v_i\right)\right\} + \left(\frac{Dx_i}{Dt}\right)_{\text{nucl.}} \Longrightarrow$$

$$\frac{Dx_i}{Dt} = \frac{\partial}{\partial m}\left\{4\pi r^2\rho\left(4\pi r^2\rho x_i\sum_{j=1}^{c} b_{ij}\frac{\partial x_j}{\partial m} + v_i x_i\right)\right\} + \left(\frac{Dx_i}{Dt}\right)_{\text{nucl.}}. \tag{6.192}$$

On a utilisé l'équation de continuité:

$$0 = \frac{\partial\rho}{\partial t} + \nabla * \rho\mathcal{V} = \frac{\partial\rho}{\partial t} + \rho\nabla * \mathcal{V} + \mathcal{V} * \nabla\rho = x_i\frac{\partial\rho}{\partial t} + x_i\rho\nabla * \mathcal{V} + x_i\mathcal{V} * \nabla\rho, \tag{6.193}$$

et la conservation des particules d'espèce i:

$$\nabla * x_i\rho\mathcal{V} = 0 \Rightarrow \rho\mathcal{V} * \nabla x_i = -x_i\rho\nabla * \mathcal{V} - x_i\mathcal{V} * \nabla\rho, \tag{6.194}$$

$$\Rightarrow \frac{\partial\rho x_i}{\partial t} = x_i\frac{\partial\rho}{\partial t} + \rho\frac{\partial x_i}{\partial t} - x_i\frac{\partial\rho}{\partial t} - x_i\rho\nabla * \mathcal{V} - x_i\mathcal{V} * \nabla\rho = \rho\left(\frac{\partial x_i}{\partial t} + \mathcal{V} * \nabla x_i\right) = \rho\frac{Dx_i}{Dt}. \tag{6.195}$$

En posant:

$$D_i \equiv (d_{i1}, \ldots, d_{ic},) \equiv -(x_i b_{i1}, \ldots, x_i b_{ic},),\ i = 1, \ldots, c, \tag{6.196}$$

on obtient la forme de l'équation de diffusion utilisée dans Cesam2k20[15]:

$$\frac{Dx_i}{Dt} = \frac{\partial F_i}{\partial m} + \Psi_i,\ F_i = 4\pi R^2\rho\left(4\pi R^2\rho D_i * \nabla_{\text{m}}\mathcal{X} + v_i x_i\right) \tag{6.197}$$

où $\nabla_{\text{m}}\mathcal{X}$ est le vecteur:

$$\left(\frac{\partial x_1}{\partial m}, \ldots, \frac{\partial x_{\text{c}}}{\partial m}\right)^{\text{T}}. \tag{6.198}$$

6.7.8 Jacobien.

Cesam2k20 intégrant l'équation de diffusion avec un schéma implicite, il est nécessaire de calculer les dérivées:

$$\frac{\partial v_i}{\partial x_k},\ \frac{\partial d_{ij}}{\partial x_k} = x_i\frac{\partial b_{ij}}{\partial x_k} + b_{ij}\delta_{ik},\ i, j, k = 1, \ldots, c,\ \delta_{ik} = 1 \text{ si } i = k,\ 0 \text{ sinon}. \tag{6.199}$$

En dérivant:

$$A^{-1}A = I \Rightarrow \frac{\partial A^{-1}}{\partial x_k}A + A^{-1}\frac{\partial A}{\partial x_k} = 0 \Rightarrow \frac{\partial A^{-1}}{\partial x_k} = -A^{-1}\frac{\partial A}{\partial x_k}A^{-1}, \tag{6.200}$$

d'où:

$$\frac{\partial A^{-1}\gamma}{\partial x_k} = \frac{\partial A^{-1}}{\partial x_k}\gamma + A^{-1}\frac{\partial\gamma}{\partial x_k} = -A^{-1}\frac{\partial A}{\partial x_k}A^{-1}\gamma + A^{-1}\frac{\partial\gamma}{\partial x_k} = A^{-1}\left(\frac{\partial\gamma}{\partial x_k} - \frac{\partial A}{\partial x_k}v\right), \tag{6.201}$$

et de façon similaire:

$$\frac{\partial B}{\partial x_k} = A^{-1}\left(\frac{\partial G}{\partial x_k} - \frac{\partial A}{\partial x_k}B\right). \tag{6.202}$$

[15]Avec la variable d'espace Lagrangienne m les opérateurs $\frac{D}{Dt}$ et $\frac{\partial}{\partial t}$ sont équivalents.

6.7.9 Accélérations radiatives

L'accélération radiative ressentie par les isotopes résulte de l'impulsion des photons qu'ils absorbent. En raison de l'anisotropie du rayonnement, même dans le milieu épais, pour la plupart des isotopes, la composante radiale de l'accélération radiative diminue la gravité locale et donc la sédimentation. Deux descriptions dues à G.Alécian sont implantées dans Cesam2k20.

On simplifie en considérant chaque isotope dans l'ensemble de ses états d'ionisation. m étant la masse à l'intérieur de la sphère de rayon r, la gravité effective $g_{\rm e}$ qui compense le gradient de pression dans l'équilibre quasi-statique, est somme de la partie purement gravitationnelle $g_{\rm G} = -Gm/r^2$, de l'accélération centrifuge moyenne $g_\Omega = \frac{2}{3} r\Omega^2$ et d'une partie radiative $g_{\rm R}$. On postule que $g_{\rm R}$ est la somme, pondérée par les masses partielles des particules ($\sum_{i\neq e} n_i m_i$) des accélérations radiatives $g_{{\rm R}i}$:

$$g_{\rm R} = \frac{\sum_{i\neq e} n_i m_i g_{{\rm R}i}}{\sum_{i\neq e} n_i m_i} = \frac{\sum_{i\neq e} \rho N_0 X_i g_{{\rm R}i}}{\sum_{i\neq e} \rho N_0 X_i} = \frac{\sum_{i\neq e} X_i g_{{\rm R}i}}{\sum_{i\neq e} X_i} = \sum_{i\neq e} x_i \nu_i g_{{\rm R}i}, \tag{6.203}$$

Sur la particule $i \neq e$, l'accélération g_i est somme de la gravité effective $g_{\rm e} \equiv g_{\rm G} + g_\Omega$ et de la partie radiative $g_{{\rm R}i}$:

$$g_i = g_{\rm e} + g_{{\rm R}i}. \tag{6.204}$$

Pour faciliter les algorithmes on pose, pour les électrons, $g_{{\rm R}e} \equiv 0$. On notera la gravité totale par:

$$g_{\rm t} \equiv g_{\rm e} + g_i = g_{\rm e} + \sum_i x_i \nu_i g_{{\rm R}i}, \;\; \frac{\partial g_{\rm t}}{\partial x_i} = \nu_j g_{{\rm R}j} + \sum_i x_i \nu_i \frac{\partial g_{{\rm R}i}}{\partial x_j}. \tag{6.205}$$

La première description du calcul des accélérations radiatives[16] est effectuée dans le cadre de l'approximation proposée par **??**. Sont seuls pris en compte les effets des transitions lié-lié. C'est une approximation semi analytique où la dépendance de l'accélération radiative à la concentration apparait dans un terme séparé de celui contenant les propriétés atomiques des ions. Cette approximation permet donc de ne calculer qu'un nombre réduit de quantités lorsque la concentration locale évolue au cours du temps. Pour l'élément $i \neq e$ l'accélération radiative est obtenue par une somme pondérée des accélérations radiatives de chaque ion j:

$$g_{{\rm R}i} = \sum_{j=0}^{Z_i} x_{j,i} g_{{\rm R}i,j}, \;\; g_{{\rm R}i,j} = g_{{\rm R}_0 i,j} \sqrt{1 + \frac{C_{j,i}}{C_{{\rm S}j,i}}}, \;\; C_{{\rm S}j,i} = b\Psi_{j,i}^2, \;\; g_{{\rm R}_0 i,j} = q_i \Phi_{j,i}, \tag{6.206}$$

où $C_{j,i} = n_{j,i}/n_{\rm H}$ est la concentration, par rapport à l'hydrogène, de l'ion j de l'élément chimique i, $C_{{\rm S}j,i}$ sa concentration de saturation et $g_{{\rm R}_0 i,j}$ l'accélération radiative à la limite de concentration nulle. On remarquera que:

$$\frac{C_{j,i}}{C_{{\rm S}j,i}} = \frac{x_{j,i}}{x_{{\rm S}j,i}}. \tag{6.207}$$

Les quantités b et q_i sont données par:

$$b = \frac{m_e m_p \mathcal{C}}{2e^2 \nu_{\rm H}} \frac{n_e \kappa}{\sqrt{T}}, \;\; q_i = \frac{\pi k^3 e^2}{2a\,\mathcal{C}^5 m_e m_u} \frac{l}{T r^2 \nu_i}, \tag{6.208}$$

$\mathcal{C}$ est la célérité de la lumière, m_p la masse du proton, κ l'opacité moyenne de Rosseland, h la constante de Plank, a la constante de la radiation, k la constante de Boltzman, r le rayon local et l la luminosité locale. Les quantités $\Psi_{j,i}^2$ et $\Phi_{j,i}$ sont tabulées dans le cas de l'approximation #2 décrite dans Alecian

[16] Ce paragraphe a été rédigé en grande partie par G. Alécian.

et al. (1993) et ce, pour des masses stellaires de $1M_\odot$ à $2M_\odot$. Les données de la base Topbase sont utilisées. Ces tables, qui correspondent à des masses différentes, sont lues au premier appel de la routine `alecian1` décrite au §7.8.1 (Page 160). Les transitions radiatives pour les isotopes d'un même élément se produisant à des fréquences très proches, l'opacité monochromatique de l'ensemble des isotopes d'un même élément est pratiquement égale à la somme des opacités monochromatiques de ses divers isotopes. Dans Cesam2k20, les isotopes apparaissent comme des éléments séparés. Pour le calcul des accélérations radiatives, les isotopes d'un même élément sont identifiés et leurs abondances sont additionnées de façon à traiter correctement l'effet de saturation des transitions lié-lié. Les accélérations radiatives sont ensuite restituées isotope par isotope.

Pour la description de la physique du second formalisme du calcul des accélérations radiatives, se référer aux publications de G.Alécian.

6.8 Diffusion du moment cinétique

Cesam2k20 offre la possibilité de traiter la diffusion du moment cinétique selon le formalisme le formalisme de Talon et al. (1997) (***tz97***) ou selon celui de Mathis & Zahn (2004) (***mz04***). Ces deux formalismes diffèrent principalement par le calcul de la fluctuation du potentiel gravitationnel, simplifié avec ***tz97***, complet avec ***mz04***. Les coefficients de diffusion peuvent être calculés suivant les prescriptions de Palacios et al. (2003) ou de Mathis et al. (2004).

La diffusion du moment cinétique et celle des espèces chimiques sont couplées par la turbulence du milieu, principalement au voisinage des limites RZ/CZ. La résolution numérique du système d'équations régissant ces deux processus de diffusion s'est avérée extrèmement instable. La solution retenue a été de résoudre séparément les sytèmes d'équations correspondant respectivement au moment cinétique et aux espèces chimiques. La méthode des éléments finis utilisée pour résoudre les équations de la diffusion du moment cinétique ne différant de celle utilisée pour la diffusion des éléments chimiques que par la prise en compte des conditions limites.

6.8.1 Changement de variable $M \rightarrow \nu \equiv (M/M_\odot)^{2/3}$

Les abondances et la vitesse angulaire étant des variables lagrangiennes, pour l'intégration des équations d'évolution de la composition chimique et du moment cinétique, on utilise la variable d'espace ν. Les transformations eulérien $\leftrightarrow$ lagrangien sont les suivantes:

$$M = M_\odot m \ ; \ \nu = m^{\frac{2}{3}} \ ; \ m = \nu^{\frac{3}{2}} \ ; \ M = M_\odot m \ ; \ d\nu = \frac{2}{3} m^{-\frac{1}{3}} dm \ ; \ dm = \frac{3}{2}\sqrt{\nu} d\nu$$

$$dM = M_\odot dm = M_\odot \frac{3}{2}\sqrt{\nu} d\nu \ ; \ \frac{\partial}{\partial M} = \frac{2}{3M_\odot\sqrt{\nu}} \frac{\partial}{\partial \nu}$$

$$R = R_\odot r \ ; \ dM = \frac{3M_\odot}{2}\sqrt{\nu} d\nu = 4\pi R^2 \rho dR = 4\pi R_\odot^2 r^2 \rho dR$$

$$dR = \frac{3M_\odot}{8\pi R_\odot^2} \frac{\sqrt{\nu}}{r^2\rho} d\nu \ ; \ \frac{\partial}{\partial R} = \frac{8\pi R_\odot^2 r^2 \rho}{3M_\odot\sqrt{\nu}} \frac{\partial}{\partial \nu}$$

6.8.2 Quelques notations

- $l = \frac{L}{L_\odot}$: luminosité,
- ν_i: masse atomique de l'élément chimique i,
- $\overline{z_i}$: charge moyenne de l'élément chimique i en tenant compte de l'ionisation,
- $\dot{\mathcal{M}}$: taux de perte de masse par unité de temps,

- $\dot{\mathcal{M}}_\Omega \leq 0$: taux de perte de moment cinétique par unité de masse et de temps.

Dans les algorithmes, les variables relatives à la rotation ont un nom ou une extension comprenant `rot`, `rota` ou encore `w`.

6.8.3 Expressions de $H_{\rm P}$, $H_{\rm T}$, ∇_μ, χ etc... et dérivées

$$H_{\rm P} = -\frac{\partial R}{\partial \ln P} = \frac{P}{\rho g} = \frac{PR^2}{GM\rho} = \frac{R_\odot^2 r^2 P}{GM_\odot \nu^{\frac{3}{2}}\rho}$$

$$\nabla = \frac{\partial \ln T}{\partial \ln P}$$

$$H_{\rm T} = -\frac{\partial R}{\partial \ln T} = -\frac{\partial R}{\partial \ln P}\frac{\partial \ln P}{\partial \ln T} = \frac{H_{\rm P}}{\nabla}$$

$$g(M,R) = \frac{GM}{R^2} = \frac{GM_\odot}{R_\odot^2}\frac{\nu^{\frac{3}{2}}}{r^2},\ \ln g = \ln G + \ln M - 2\ln R \Rightarrow \frac{\partial \ln g}{\partial \ln R} = \frac{\partial \ln M}{\partial \ln R} - 2 = \frac{4\pi R_\odot^3}{M_\odot}\frac{r^3\rho}{\nu^{\frac{3}{2}}} - 2$$

$$\frac{\tilde{g}}{g} = \frac{4R^3}{3GM}\Omega^2 = \frac{4R_\odot^3}{3GM_\odot}\frac{r^3}{\nu^{\frac{3}{2}}}\Omega^2$$

$$x_i = \frac{X_i}{\nu_i},\ \mu^{-1} = \sum_i (1+\overline{z_i})x_i,\ \frac{\partial \mu}{\partial x_i} = -\mu^2(1+\overline{z_i})$$

$$\mu^{-1} \simeq \frac{1}{16}(20X + 12 - 3Z) = \frac{1}{16}(20X + 12 - 3(1-X-Y)) = \frac{1}{16}(23X + 3Y + 9) =$$

$$\frac{1}{16}(23\nu_{\rm H}x_{\rm H} + 3\nu_{\rm He}x_{\rm He} + 9)$$

$$\mu^{-1} \simeq 2X + \frac{3}{4}Y + \frac{9}{16}Z = \frac{1}{16}\left[32X + 12\left(1-X-Y\right) + 9Z\right] = \frac{1}{16}(20X + 12 - 3Z)$$

$$\mu = \frac{2}{1 + 2X + 0.5Y} = \frac{2}{1.5 + 2.5X - 0.5Z}$$

$$\frac{\partial \ln \mu}{\partial \nu} = \frac{1}{\mu}\sum_i \frac{\partial \mu}{\partial x_i}\frac{\partial x_i}{\partial \nu} = -\mu\sum_i (1+\overline{z_i})\frac{\partial x_i}{\partial \nu} \simeq -\frac{\mu}{16}\left(23\nu_{\rm H}\frac{\partial x_{\rm H}}{\partial \nu} + 3\nu_{\rm He}\frac{\partial x_{\rm He}}{\partial \nu}\right)$$

$$\phi = -\frac{\mu}{80C_{\rm P}}\left\{(12Z-48)\ln\left(64 - 48\mu + 12Z\mu\right) + (32-23Z)\ln\left(-16 + 32\mu - 23Z\mu\right)\right.$$

$$\left.+5Z\ln\left(5Z\right) + (16+6Z)\ln\left(32 + 16\mu + 6Z\mu\right)\right\}$$

$$\varepsilon(T,\mu) \Rightarrow \varepsilon_\mu = \frac{\partial \ln \varepsilon}{\partial \ln \mu} = \frac{\mu}{\varepsilon}\sum_i \frac{\partial \varepsilon}{\partial x_i}\frac{\partial x_i}{\partial \mu} = -\frac{1}{\mu\varepsilon}\sum_i \frac{1}{1+\overline{z_i}}\frac{\partial \varepsilon}{\partial x_i} = \frac{\rho}{\varepsilon}\frac{\partial \varepsilon}{\partial \rho}\frac{\partial \ln \rho}{\partial \ln \mu} = \varphi\frac{\rho}{\varepsilon}\frac{\partial \varepsilon}{\partial \rho}$$

$$\frac{\partial \ln \mu}{\partial X} \simeq \frac{1}{\mu}\frac{\partial \mu}{\partial X} = -\frac{23\mu}{16},\ \frac{\partial \ln \mu}{\partial Y} \simeq -\frac{3\mu}{16}$$

$$\varepsilon_\mu \simeq \frac{1}{\varepsilon}\left(\frac{\partial \varepsilon}{\partial X}\frac{\partial X}{\partial \ln \mu} + \frac{\partial \varepsilon}{\partial Y}\frac{\partial Y}{\partial \ln \mu}\right) = -\frac{16}{\varepsilon\mu}\left(\frac{1}{23\nu_{\rm H}}\frac{\partial \varepsilon}{\partial x_{\rm H}} + \frac{1}{3\nu_{\rm He}}\frac{\partial \varepsilon}{\partial x_{\rm He}}\right)$$

$$\varepsilon_{\rm T} = \frac{\partial \ln \varepsilon}{\partial \ln T} = \frac{T}{\varepsilon}\frac{\partial \varepsilon}{\partial T}$$

$$\varphi = \frac{\partial \ln \rho}{\partial \ln \mu} = \frac{\mu}{\rho}\frac{\partial \rho}{\partial \mu} = \frac{\mu}{\rho}\sum_i \frac{\partial \rho}{\partial X_i}\frac{\partial X_i}{\partial \mu} = -\frac{1}{\mu\rho}\sum_i \frac{\nu_i}{1+\overline{z_i}}\frac{\partial \rho}{\partial X_i}$$

$$\mathrm{d}\ln \rho = \alpha \mathrm{d}\ln P - \delta \mathrm{d}\ln T + \varphi \mathrm{d}\ln \mu = (\alpha - \delta\nabla)\,\mathrm{d}\ln P + \varphi \mathrm{d}\ln \mu,$$

$$\frac{\partial P}{\partial R} = -\frac{GM\rho}{R^2} \Rightarrow \frac{\partial P}{\partial \nu} = -\frac{3GM_\odot^2}{8\pi R_\odot^4}\frac{\nu^2}{r^4} \Rightarrow \frac{\partial \ln P}{\partial \nu} = -\frac{3GM_\odot^2}{8\pi R_\odot^4}\frac{\nu^2}{r^4 P} \Rightarrow$$

$$\frac{\partial \ln \rho}{\partial \nu} = -\frac{3GM_\odot^2}{8\pi R_\odot^4}\frac{(\alpha - \delta\nabla)\,\nu^2}{r^4 P} + \varphi\frac{\partial \ln \mu}{\partial \nu}$$

$$\rho_m = \frac{3M}{4\pi R^3} = \frac{3M_\odot}{4\pi R_\odot^3}\frac{\nu^{\frac{3}{2}}}{r^3}$$

$$\text{si } \frac{\partial \ln \mu}{\partial \nu} \neq 0,\ \varphi = \left(\frac{\partial \ln \rho}{\partial \nu} - \alpha\frac{\partial \ln P}{\partial \nu} + \delta\frac{\partial \ln T}{\partial \nu}\right) \Big/ \left(\frac{\partial \ln \mu}{\partial \nu}\right)$$

$$\nabla_\mu = \frac{\partial \ln \mu}{\partial \ln P} = \frac{\partial \ln \mu}{\partial R}\frac{\partial R}{\partial \ln P} = -\frac{\partial \ln \mu}{\partial R}H_{\rm P} = -\frac{8\pi R_\odot^2 r^2 \rho}{3M_\odot\sqrt{\nu}}H_{\rm P}\frac{\partial \ln \mu}{\partial \nu} =$$

$$\frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho\mu H_{\rm P}}{\sqrt{\nu}}\sum_i (1+\overline{z_i})\frac{\partial x_i}{\partial \nu} \simeq \frac{\pi R_\odot^2 r^2 \rho}{6M_\odot\sqrt{\nu}}H_{\rm P}\left(23\nu_{\rm H}\frac{\partial x_{\rm H}}{\partial \nu} + 3\nu_{\rm He}\frac{\partial x_{\rm He}}{\partial \nu}\right)$$

$$\nabla_{\rm rad} = \frac{3\kappa LP}{16\pi acGMT^4} = \frac{3L_\odot}{16\pi acGM_\odot}\frac{\kappa l P}{\nu^{\frac{3}{2}}T^4}$$

$$N_{\rm T}^2 = \frac{g\delta}{H_{\rm P}}(\nabla_{\rm ad} - \nabla),\quad N^2 = \frac{g\varphi}{H_{\rm P}}\nabla$$

6.8.4 Les coefficients de diffusion $D_{\rm h}$, $D_{\rm v}$ et $D_{\rm eff}$

Formalisme de Mathis, Palacios & Zahn

Les coefficients de diffusivité horizontale $D_{\rm h}$ et verticale $D_{\rm v}$ sont donnés par Mathis et al. (2004, eq. 7-10-19):

$$D_{\rm h} = R\sqrt{\beta\Omega R\,|2V-\alpha U|},\ \beta = 1.5\,10^{-6},\ V = \frac{1}{6\rho R}\frac{\partial R^2\rho U}{\partial R},\ \alpha = \frac{1}{2}\frac{\partial \ln R^2\Omega}{\partial \ln R}$$

$$D_{\rm h}^2 = R^3\beta\Omega\left|\frac{1}{3R\rho}\frac{\partial R^2\rho U}{\partial R} - \frac{RU}{2R^2\Omega}\frac{\partial R^2\Omega}{\partial R}\right| = R^2\beta\Omega\left|\frac{1}{3\rho}\frac{\partial R^2\rho U}{\partial R} - \frac{U}{2\Omega}\frac{\partial R^2\Omega}{\partial R}\right| =$$

$$R^2\beta\Omega\left|\frac{R^2\rho}{3\rho}\frac{\partial U}{\partial R} + \frac{2R\rho U}{3\rho} + \frac{R^2U}{3\rho}\frac{\partial \rho}{\partial R} - \frac{2URR\Omega}{2\Omega} - \frac{UR^2}{2\Omega}\frac{\partial \Omega}{\partial R}\right| =$$

$$R^2\beta\Omega\left|\frac{R^2}{3}\frac{\partial U}{\partial R} + \frac{2}{3}UR + \frac{R^2U}{3\rho}\frac{\partial \rho}{\partial R} - UR - \frac{UR^2}{2\Omega}\frac{\partial \Omega}{\partial R}\right| =$$

$$R^4\beta\Omega\left|\frac{1}{3}\frac{\partial U}{\partial R} - \frac{U}{3R} + \frac{U}{3\rho}\frac{\partial \rho}{\partial R} - \frac{U}{2\Omega}\frac{\partial \Omega}{\partial R}\right| \Rightarrow$$

$$D_{\rm h}^2 = \frac{R_\odot^4\beta}{3}r^4\Omega\left|\frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho}{\sqrt{\nu}}\left[\frac{\partial U}{\partial \nu} + U\left(\frac{\partial \ln \rho}{\partial \nu} - \frac{3}{2\Omega}\frac{\partial \Omega}{\partial \nu}\right)\right] - \frac{U}{R_\odot r}\right|$$

$$D_{\rm h}^2 = \left|C_1\Omega\frac{\partial U}{\partial \nu} + C_2\Omega U - C_3U\frac{\partial \Omega}{\partial \nu}\right|,\quad C_1 = \frac{8\pi R_\odot^6\beta}{9M_\odot}\frac{r^6\rho}{\sqrt{\nu}}$$

$$C_2 = C_1^* - C_2^*,\quad C_1^* = \frac{8\pi R_\odot^6\beta}{9M_\odot}\frac{r^6\rho}{\sqrt{\nu}}\frac{\partial \ln \rho}{\partial \nu},\quad C_2^* = \frac{R_\odot^3\beta}{3}r^3,\quad C_3 = \frac{4\pi R_\odot^6\beta}{3M_\odot}\frac{r^6\rho}{\sqrt{\nu}}$$

$$D_{\rm v} = \frac{Ri_c(K+D_{\rm h})R^2}{N_{\rm T}^2 + N_\mu^2(1+K/D_{\rm h})}\left(\frac{\partial \Omega}{\partial R}\right)^2 = \frac{64\pi^2 Ri_cR_\odot^6}{9M_\odot^2}\frac{r^6\rho^2}{\nu}\frac{D_{\rm h}(1+\frac{K}{D_{\rm h}})}{\left[N_{\rm T}^2 + N_\mu^2\left(1+\frac{K}{D_{\rm h}}\right)\right]}\left(\frac{\partial \Omega}{\partial \nu}\right)^2$$

$$D_{\rm v} = \begin{cases} \nu_{\rm cin} & \text{si } D_{\rm v} < \nu_{\rm cin} \\ D_{\rm v} + \nu_{\rm cin} & \text{sinon} \end{cases},\ \nu_{\rm cin} = \frac{2\,10^{-15}T^{\frac{5}{2}}\sqrt{\bar{\nu}}}{\rho\bar{z}^4\ln\lambda},\ \lambda = 1.3\,10^4T^{\frac{3}{2}}n_e^{-\frac{1}{2}}$$

$$Ri_c = \frac{1}{6},\ \bar{z} = \frac{\sum_i \bar{z}_ix_i}{\sum_i x_i},\ \bar{\nu} = \frac{\sum_i x_i\nu_i}{\sum_i x_i}$$

$$D_{\rm eff} = \frac{R_\odot^2r^2U^2}{30D_{\rm h}} + D_{\rm v}$$

Lorsque le diffusion microscopique est prise en compte, le coefficient $D_{\rm eff}$ de diffusion turbulente généré par la diffusion du moment cinétique est ajouté au coefficient de diffusion turbulente.

Formalisme simplifié de Castro, Vauclair & Richard

Castro et al. (2007, eq. 3-4-5) ont utilisé des coefficients de diffusivité simplifiés:

$$D_{\rm h} = C_{\rm h}R|U|,\ C_{\rm h} = 5\times10^4$$

$$D_{\rm v} = C_{\rm v}R|U|,\ C_{\rm v} = \alpha - \frac{1}{30C_{\rm h}},\ \alpha = 6$$

$$D_{\rm eff} = \alpha R|U|$$

6.8.5 Formalisme de Talon et al. (1997)

Les variables

- $x_i,\ i = 1,\ldots,n_c$: abondances par mole

- $y_1 \equiv \Omega$: vitesse angulaire s^{-1},
- $y_2 \equiv U$: vitesse de circulation méridienne,
- $y_3 \equiv \Theta = \frac{\tilde{\rho}}{\rho}$: taux de fluctuation de la densité le long d'une isobare
- $y_4 \equiv \Lambda = \frac{\tilde{\mu}}{\mu}$: taux de fluctuation du poids moléculaire moyen le long d'une isobare
- $y_5 \equiv \Psi \equiv \frac{\tilde{T}}{T} = \frac{\varphi\Lambda - \Theta}{\delta}$: taux de fluctuation de la température le long d'une isobare
- $y_6 \equiv \mathcal{T}$: taux de transfert du moment cinétique par unité de masse.
- $y_7 \equiv \Upsilon$: second membre de l'équation de la vitesse de circulation méridienne.

Pour l'intégration numérique, le système des équations de la diffusion du moment cinétique est transformé en un sytème de 7 équations du premier ordre. La variable indépendante lagrangienne est ν.

Transport du moment cinétique

Dans une zone radiative, le moment cinétique par unité de masse, $R^2\Omega = R^2 y_1$, vérifie l'équation de transport (Talon et al., 1997, eq. 4):

$$\rho\frac{DR^2\Omega}{Dt} = \frac{1}{R^2}\frac{\partial}{\partial R}\left[R^4\rho\left(\frac{\Omega U}{5} + D_{\rm v}\frac{\partial\Omega}{\partial R}\right)\right] - \rho\dot{\mathcal{M}}_\Omega \Rightarrow \tag{6.209}$$

$$R_\odot^2\frac{Dr^2\Omega}{Dt} = \frac{8\pi R_\odot^6 r^2\rho}{3M_\odot R_\odot^2 r^2\rho\sqrt{\nu}}\frac{\partial}{\partial\nu}\left[r^4\rho\left(\frac{\Omega U}{5} + \frac{8\pi R_\odot^2 r^2\rho D_{\rm v}}{3M_\odot\sqrt{\nu}}\frac{\partial\Omega}{\partial\nu}\right)\right] - \dot{\mathcal{M}}_\Omega \Rightarrow$$

$$\frac{Dr^2\Omega}{Dt} = \frac{8\pi R_\odot^2}{3M_\odot\sqrt{\nu}}\frac{\partial}{\partial\nu}\left[r^4\rho\left(\frac{\Omega U}{5} + \frac{8\pi R_\odot^2 r^2\rho}{3M_\odot\sqrt{\nu}}D_{\rm v}\frac{\partial\Omega}{\partial\nu}\right)\right] - \frac{\dot{\mathcal{M}}_\Omega}{R_\odot^2} \Rightarrow$$

$$\sqrt{\nu}\frac{Dr^2\Omega}{Dt} = \frac{8\pi R_\odot^2}{15M_\odot}\frac{\partial}{\partial\nu}(r^4\rho\Omega U) + \frac{64\pi^2 R_\odot^4}{9M_\odot^2}\frac{\partial}{\partial\nu}\left(\frac{r^6\rho^2 D_{\rm v}}{\sqrt{\nu}}\frac{\partial\Omega}{\partial\nu}\right) - \frac{\dot{\mathcal{M}}_\Omega\sqrt{\nu}}{R_\odot^2}$$

Pour l'intégration par collocation les équations sont mises sous la forme :

$$C_1\Omega - C_2 = \frac{\partial\mathcal{T}}{\partial\nu} \tag{6.210}$$

$$\mathcal{T} = C_3\Omega U + C_4\frac{\partial\Omega}{\partial\nu} \tag{6.211}$$

$$C_1 = \frac{r^2\sqrt{\nu}}{\Delta t},\ C_2 = (C_{11}^* - C_{12}^*)\sqrt{\nu},\ C_{11}^* = \frac{r^2(t)\Omega(t)}{\Delta t},\ C_{12}^* = \frac{\dot{\mathcal{M}}_\Omega}{R_\odot^2}$$

$$C_3 = \frac{8\pi R_\odot^2}{15M_\odot}r^4\rho,\ C_4 = \frac{64\pi^2 R_\odot^4}{9M_\odot^2}\frac{r^6\rho^2 D_{\rm v}}{\sqrt{\nu}}$$

Expressions initiales de $\breve{E}_\Omega$ et $\breve{E}_\mu$

Au paragraphe suivant, dans les expressions initiales de Talon et al. (1997), pour E_Ω et E_μ des termes ont été regroupés. Pour référence, on reproduit les équations originales notées $\breve{E}_\Omega$ pour $\tilde{E}_\Omega$ et $\breve{E}_\mu$ pour $\tilde{E}_\mu$; $\tilde{E}_\Omega$ et $\tilde{E}_\mu$ sont définis plus avant.

$$\begin{aligned}\breve{E}_\Omega = \quad & -\frac{\rho_m}{\rho}\left\{\frac{R}{3}\frac{\partial}{\partial R}\left[H_{\rm T}\frac{\partial}{\partial R}\left(\frac{\Theta}{\delta}\right) + \left(1 - \frac{\chi_{\rm T}+1}{\delta}\right)\Theta\right] - \frac{2H_{\rm T}}{R}\left(1 + \frac{D_{\rm h}}{K}\right)\frac{\Theta}{\delta} + \frac{2}{3}\Theta\right\} \\ & -\frac{\bar{\varepsilon}+\bar{\varepsilon}_g}{\varepsilon_{\rm m}}\left\{H_{\rm T}\frac{\partial}{\partial R}\left(\frac{\Theta}{\delta}\right) + \left[f_\varepsilon\left(\frac{\varepsilon_{\rm T}}{\delta} - 1\right) + 2 - \frac{\chi_{\rm T}+1}{\delta}\right]\Theta\right\}\end{aligned}$$

$$
\begin{aligned}
\breve{E}_\mu = \quad & \frac{\rho_m}{\rho}\left\{\frac{R}{3}\frac{\partial}{\partial R}\left[H_{\mathrm{T}}\frac{\partial}{\partial R}\left(\frac{\varphi}{\delta}\Lambda\right)-\left(\chi_\mu+\frac{\chi_{\mathrm{T}}+1}{\delta}\varphi\right)\Lambda\right]-\frac{2H_{\mathrm{T}}}{R}\frac{\varphi}{\delta}\Lambda\right\} \\
& +\frac{\bar{\varepsilon}+\bar{\varepsilon}_g}{\varepsilon_{\mathrm{m}}}\left[H_{\mathrm{T}}\frac{\partial}{\partial R}\left(\frac{\varphi}{\delta}\Lambda\right)+\left(f_\varepsilon\varepsilon_\mu+f_\varepsilon\varepsilon_{\mathrm{T}}\frac{\varphi}{\delta}-\chi_\mu-\frac{\chi_{\mathrm{T}}+1}{\delta}\varphi\right)\Lambda\right]
\end{aligned}
$$

Expression vérifiée par E_Ω

Les quantités concernées[17] de $\breve{E}_\Omega$ et $\breve{E}_\mu$ ont été regroupées dans E_Ω.

$$
\begin{aligned}
E_\Omega = \quad & 2\left(1-\frac{\Omega^2}{2\pi G\rho}+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\bar{\varepsilon}+\bar{\varepsilon}_g}{\varepsilon_{\mathrm{m}}}\right)\frac{\tilde{g}}{g}+\tilde{E_\Omega} \\
= \quad & \frac{8R_\odot^3}{3GM_\odot}\left(1-\frac{\Omega^2}{2\pi G\rho}+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\partial\ln l}{\partial\ln m}\right)\frac{r^3\Omega^2}{\nu^{\frac{3}{2}}}+\tilde{E_\Omega} \\
\tilde{E_\Omega} = \quad & \frac{\rho_m}{\rho}\left\{\frac{R}{3}\frac{\partial}{\partial R}\left[H_{\mathrm{T}}\frac{\partial}{\partial R}\left(\frac{\varphi\Lambda-\Theta}{\delta}\right)\right.\right. \\
& \left.-(1+\chi_{\mathrm{T}})\frac{\varphi\Lambda-\Theta}{\delta}-\Theta-\chi_\mu\Lambda\right] \\
& \left.-2\frac{H_{\mathrm{T}}}{R}\left(\frac{\varphi\Lambda-\Theta}{\delta}+\frac{R\Theta}{3H_{\mathrm{T}}}-\frac{D_{\mathrm{h}}}{K}\frac{\Theta}{\delta}\right)\right\} \\
\tilde{E_\Omega} = \quad & \frac{3M_\odot}{4\pi R_\odot^3}\frac{\nu^{\frac{3}{2}}}{r^3\rho}\left\{\frac{8\pi R_\odot^3}{9M_\odot}\frac{r^3\rho}{\sqrt{\nu}}\frac{\partial}{\partial\nu}\left[\frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho H_{\mathrm{T}}}{\sqrt{\nu}}\frac{\partial}{\partial\nu}\Psi\right.\right. \\
& \left.-(1+\chi_{\mathrm{T}})\Psi-\Theta-\chi_\mu\Lambda\right] \\
& \left.-2\frac{H_{\mathrm{T}}}{R_\odot r}\left(\Psi+\frac{R_\odot}{3}\frac{r\Theta}{H_{\mathrm{T}}}-\frac{D_{\mathrm{h}}}{K}\frac{\Theta}{\delta}\right)\right\} \\
\tilde{E_\Omega} = \quad & \nu\frac{16\pi R_\odot^2}{9M_\odot}\frac{\partial}{\partial\nu}\left(\frac{r^2\rho H_{\mathrm{T}}}{\sqrt{\nu}}\frac{\partial\Psi}{\partial\nu}\right) \\
& +\frac{2}{3}\nu\frac{\partial}{\partial\nu}\left\{-(1+\chi_{\mathrm{T}})\Psi-\Theta-\chi_\mu\Lambda\right\} \\
& -\nu\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\mathrm{T}}\sqrt{\nu}}{r^4\rho}\left[\Psi+\left(\frac{R_\odot r}{3H_{\mathrm{T}}}-\frac{D_{\mathrm{h}}}{K\delta}\right)\Theta\right]
\end{aligned}
$$

Expression vérifiée par E_μ

Les quantités concernées de $\breve{E}_\Omega$ et $\breve{E}_\mu$ ont été regroupées dans E_μ.

$$
\begin{aligned}
E_\mu = \quad & \frac{\bar{\varepsilon}+\bar{\varepsilon}_g}{\varepsilon_{\mathrm{m}}}\left[H_{\mathrm{T}}\frac{\partial}{\partial R}\left(\frac{\varphi\Lambda-\Theta}{\delta}\right)\right. \\
& \left.+(f_\varepsilon\varepsilon_{\mathrm{T}}-\chi_{\mathrm{T}}-1)\frac{\varphi\Lambda-\Theta}{\delta}-(2-f_\varepsilon)\Theta+(f_\varepsilon\varepsilon_\mu-\chi_\mu)\Lambda\right]= \\
& \nu\frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\mathrm{T}}}{\nu^{\frac{3}{2}}}\frac{\partial\Psi}{\partial\nu}+\nu\frac{M_\odot}{L_\odot}\frac{\varepsilon_{nuc}}{l}\sqrt{\nu}\left(\varepsilon_{\mathrm{T}}\Psi+\Theta+\varepsilon_\mu\Lambda\right) \\
& -\nu\frac{\partial\ln l}{\partial\ln m}\frac{1}{\nu}\left[(\chi_{\mathrm{T}}+1)\Psi+2\Theta+\chi_\mu\Lambda\right]
\end{aligned}
$$

[17] La quantité $\frac{\Omega_0^2}{2\pi G\rho_0}$ a été introduite par S.Mathis & JP. Zahn, pour éliminer une singularité sur U au centre, quand le cœur est radiatif.

Expression vérifiée par $U = y_2$

Dans les zones radiatives:

$$\frac{M}{L}\left[\frac{Uc_pT}{H_p}\left(\nabla_{ad}-\nabla+\frac{\varphi}{\delta}\nabla_\mu\right)-\frac{c_pT}{\delta}\frac{D\Theta}{Dt}\right] \tag{6.212}$$

$$-2\left(1-\frac{\Omega^2}{2\pi G\rho}+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\bar{\varepsilon}+\bar{\varepsilon}_g}{\varepsilon_{\rm m}}\right)\frac{\tilde{g}}{g}=\tilde{E_\Omega}+E_\mu \Rightarrow \tag{6.213}$$

$$\frac{M_\odot}{L_\odot}\frac{\nu^{\frac{3}{2}}}{l}\left[\frac{c_pT}{H_p}\left(\nabla_{ad}-\nabla+\frac{\varphi}{\delta}\nabla_\mu\right)U-\frac{c_pT}{\delta}\frac{\Theta-\Theta(t)}{\Delta t}\right]$$

$$-\frac{8R_\odot^3}{3GM_\odot}\left(1-\frac{\Omega^2}{2\pi G\rho}+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\partial\ln l}{\partial\ln m}\right)\frac{r^3\Omega^2}{\nu^{\frac{3}{2}}}=$$

$$\nu\left\{\frac{16\pi R_\odot^2}{9M_\odot}\frac{\partial}{\partial\nu}\left(\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial\Psi}{\partial\nu}\right)-\frac{2}{3}\frac{\partial}{\partial\nu}\left[\left(1+\chi_{\rm T}\right)\Psi+\Theta+\chi_\mu\Lambda\right]\right.$$

$$+\frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial\Psi}{\partial\nu}-\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}\left[\Psi+\left(\frac{R_\odot r}{3H_{\rm T}}-\frac{D_{\rm h}}{K\delta}\right)\Theta\right]$$

$$\left.+\frac{M_\odot}{L_\odot}\frac{\varepsilon_{nuc}}{l}\sqrt{\nu}\left(\varepsilon_{\rm T}\Psi+\Theta+\varepsilon_\mu\Lambda\right)-\frac{\partial\ln l}{\partial\ln m}\frac{1}{\nu}\left[\left(\chi_{\rm T}+1\right)\Psi+2\Theta+\chi_\mu\Lambda\right]\right\}$$

$$\Longrightarrow$$

$$\frac{M_\odot}{L_\odot}\frac{\nu^{\frac{3}{2}}}{l\nu}\left[\frac{c_pT}{H_p}\left(\nabla_{ad}-\nabla+\frac{\varphi}{\delta}\nabla_\mu\right)U-\frac{c_pT}{\delta}\frac{\Theta-\Theta(t)}{\Delta t}\right]$$

$$-\frac{8R_\odot^3}{3GM_\odot}\left(1-\frac{\Omega^2}{2\pi G\rho}+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\partial\ln l}{\partial\ln m}\right)\frac{r^3\Omega^2}{\nu^{\frac{5}{2}}}-\frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial\Psi}{\partial\nu}$$

$$+\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}\left[\Psi+\left(\frac{R_\odot r}{3H_{\rm T}}-\frac{D_{\rm h}}{K\delta}\right)\Theta\right]-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{nuc}}{l}\sqrt{\nu}\left(\varepsilon_{\rm T}\Psi+\Theta+\varepsilon_\mu\Lambda\right)$$

$$+\frac{\partial\ln l}{\partial\ln m}\frac{1}{\nu}\left[\left(\chi_{\rm T}+1\right)\Psi+2\Theta+\chi_\mu\Lambda\right]=\frac{\partial}{\partial\nu}\left(\frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial\Psi}{\partial\nu}-\frac{2}{3}\left[\left(1+\chi_{\rm T}\right)\Psi+\Theta+\chi_\mu\Lambda\right]\right)$$

$$\Longrightarrow$$

$$\frac{M_\odot}{L_\odot}\frac{c_pT\sqrt{\nu}}{lH_p}\left(\nabla_{ad}-\nabla+\frac{\varphi}{\delta}\nabla_\mu\right)U-\frac{M_\odot}{L_\odot}\frac{c_pT\sqrt{\nu}}{l\delta}\frac{\Theta-\Theta(t)}{\Delta t}$$

$$-\frac{8R_\odot^3}{3GM_\odot}\frac{r^3}{\nu^{\frac{5}{2}}}\Omega^2+\frac{4R_\odot^3}{3\pi G^2M_\odot}\frac{r^3}{\rho\nu^{\frac{5}{2}}}\Omega^4-\frac{4R_\odot^3\Omega_0^2}{3\pi G^2M_\odot\rho_0}\frac{r^3}{\nu^{\frac{5}{2}}}\Omega^2+\frac{8R_\odot^3}{3GM_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^3}{\nu^{\frac{5}{2}}}\Omega^2$$

$$+\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}\Psi-\frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial\Psi}{\partial\nu}+\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}\left(\frac{R_\odot}{3}\frac{r}{H_{\rm T}}-\frac{D_{\rm h}}{K\delta}\right)\Theta$$

$$-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_{\rm T}\sqrt{\nu}}{l}\Psi-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\sqrt{\nu}}{l}\Theta-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_\mu\sqrt{\nu}}{l}\Lambda+\frac{\partial\ln l}{\partial\ln m}\frac{\chi_{\rm T}+1}{\nu}\Psi$$

$$+\frac{2}{\nu}\frac{\partial\ln l}{\partial\ln m}\Theta+\frac{\partial\ln l}{\partial\ln m}\frac{\chi_\mu}{\nu}\Lambda=\frac{\partial}{\partial\nu}\left(\frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial\Psi}{\partial\nu}-\frac{2}{3}\left[\left(1+\chi_{\rm T}\right)\Psi+\Theta+\chi_\mu\Lambda\right]\right)$$

$$\Longrightarrow$$

$$\frac{4R_\odot^3}{3\pi G^2M_\odot}\frac{r^3}{\rho\nu^{\frac{5}{2}}}\Omega^4-\frac{8R_\odot^3}{3GM_\odot}\frac{r^3}{\nu^{\frac{5}{2}}}\left(1+\frac{\Omega_0^2}{2\pi G\rho_0}-\frac{\partial\ln l}{\partial\ln m}\right)\Omega^2$$

$$+\frac{M_\odot}{L_\odot}\frac{c_pT\sqrt{\nu}}{lH_p}\left(\nabla_{ad}-\nabla+\frac{\varphi}{\delta}\nabla_\mu\right)U-\frac{M_\odot}{L_\odot}\frac{c_pT\sqrt{\nu}}{l\delta\Delta t}\Theta+\frac{M_\odot}{L_\odot}\frac{c_pT\sqrt{\nu}\Theta(t)}{l\delta\Delta t}$$

$$+\left(\frac{M_\odot}{2\pi R_\odot^3}\frac{\sqrt{\nu}}{r^3\rho}-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\sqrt{\nu}}{l}+\frac{2}{\nu}\frac{\partial\ln l}{\partial\ln m}-\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho K\delta}D_{\rm h}\right)\Theta$$

$$+\left(\frac{\partial\ln l}{\partial\ln m}\frac{\chi_\mu}{\nu}-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_\mu\sqrt{\nu}}{l}\right)\Lambda+\left(\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}-\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_{\rm T}\sqrt{\nu}}{l}+\frac{\partial\ln l}{\partial\ln m}\frac{\chi_{\rm T}+1}{\nu}\right)\Psi$$

$$-\frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial\Psi}{\partial\nu}=\frac{\partial}{\partial\nu}\left(\frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial\Psi}{\partial\nu}-\frac{2}{3}\left[\left(1+\chi_{\rm T}\right)\Psi+\Theta+\chi_\mu\Lambda\right]\right)$$

La forme utilisée pour l'intégration numérique est :

$$\left(C_5\Omega^2 - C_7\right)\Omega^2 + C_{10} + C_6 U + C_{11}\Theta + C_{15}\Lambda + C_{12}\Psi - C_{13}\frac{\partial\Psi}{\partial\nu} = \frac{\partial\Upsilon}{\partial\nu} \tag{6.214}$$

$$\Upsilon = C_{17}\Theta + C_{18}\Lambda + C_{16}\Psi + C_{14}\frac{\partial\Psi}{\partial\nu} \tag{6.215}$$

$$C_5 = \frac{4R_\odot^3}{3\pi G^2 M_\odot}\frac{r^3}{\rho\nu^{\frac{5}{2}}}, \quad C_6 = \frac{M_\odot}{L_\odot}\frac{c_p T\sqrt{\nu}}{lH_p}\left|\nabla_{ad} - \nabla + \frac{\varphi}{\delta}\nabla_\mu\right|,$$

$$C_7 = \frac{8R_\odot^3}{3GM_\odot}\frac{r^3}{\nu^{\frac{5}{2}}}\left(1 + \frac{\Omega_0^2}{2\pi G\rho_0} - \frac{\partial\ln l}{\partial\ln m}\right), \quad C_{10} = \frac{M_\odot}{L_\odot}\frac{c_p T\sqrt{\nu}}{l\delta\Delta t}\Theta(t)$$

$$C_{11} = C_2^* - C_3^* + C_4^* - C_5^* - C_{15}^*, \quad C_2^* = \frac{M_\odot}{2\pi R_\odot^3}\frac{\sqrt{\nu}}{r^3\rho}, \quad C_3^* = \frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\sqrt{\nu}}{l}$$

$$C_4^* = \frac{2}{\nu}\frac{\partial\ln l}{\partial\ln m}, \quad C_5^* = \frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}D_{\rm h}}{r^4\rho K\delta}, \quad C_{15}^* = \frac{M_\odot}{L_\odot}\frac{c_p T\sqrt{\nu}}{l\delta\Delta t}$$

$$C_{12} = C_6^* - C_7^* + C_8^*, \quad C_6^* = \frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\rm T}\sqrt{\nu}}{r^4\rho}, \quad C_7^* = \frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_{\rm T}\sqrt{\nu}}{l}$$

$$C_8^* = \frac{\partial\ln l}{\partial\ln m}\frac{\chi_{\rm T}+1}{\nu}, \quad C_{13} = \frac{8\pi R_\odot^2}{3M_\odot}\frac{\partial\ln l}{\partial\ln m}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}, \quad C_{14} = \frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}$$

$$C_{15} = C_9^* - C_{10}^*, \quad C_9^* = \frac{\partial\ln l}{\partial\ln m}\frac{\chi_\mu}{\nu}, \quad C_{10}^* = \frac{M_\odot}{L_\odot}\frac{\varepsilon_{\rm nuc}\varepsilon_\mu\sqrt{\nu}}{l}$$

$$C_{16} = -\frac{2}{3}\left(1+\chi_{\rm T}\right), \quad C_{17} = -\frac{2}{3}, \quad C_{18} = -\frac{2}{3}\chi_\mu$$

Expression vérifiée par $\Theta = y_3$

$$\Theta = \frac{\tilde{\rho}}{\bar{\rho}} = \frac{R^2}{3g}\frac{\partial\Omega^2}{\partial R} \tag{6.216}$$

$$\Theta = \frac{R_\odot^2 r^2 R_\odot^2 r^2 8\pi R_\odot^2 r^2\rho 2\Omega}{3GM_\odot\nu^{\frac{3}{2}}3M_\odot\sqrt{\nu}}\frac{\partial\Omega}{\partial\nu} = \frac{16\pi R_\odot^6}{9GM_\odot^2}\frac{r^6\rho\Omega}{\nu^2}\frac{\partial\Omega}{\partial\nu}$$

La forme utilisée pour l'intégration numérique est :

$$\Theta - C_{19}\Omega\frac{\partial\Omega}{\partial\nu} = 0 \tag{6.217}$$

$$C_{19} = \frac{16\pi R_\odot^6}{9GM_\odot^2}\frac{r^6\rho}{\nu^2}$$

Fluctuation du poids moléculaire $\Lambda = y_4$

L'expression vérifiée par Λ est un problème de valeur initiale:

$$\frac{D\Lambda}{Dt} = \frac{\nabla_\mu U}{H_{\rm P}} - \frac{6D_{\rm h}\Lambda}{R^2}$$

La forme utilisée pour l'intégration numérique est :

$$C_{20}\Lambda - C_{22} - C_{21}U = 0 \Longleftrightarrow \Lambda = \frac{C_{22} + C_{21}U}{C_{20}} \tag{6.218}$$

$$C_{20} = C_1^* + C_{13}^*, \quad C_1^* = \frac{1}{\Delta t}, \quad C_{13}^* = \frac{6D_{\rm h}}{R_\odot^2 r^2}, \quad C_{21} = \frac{\nabla_\mu}{H_{\rm P}}, \quad C_{22} = \frac{\Lambda(t)}{\Delta t}$$

Expression susbtituée à Λ dans les équations précédentes.

Expression vérifiée par $\Psi = y_5$

$$\Psi = \frac{\varphi\Lambda - \Theta}{\delta}$$

La forme utilisée pour l'intégration numérique est :

$$\Psi - C_{24}\Lambda + C_{23}\Theta = 0 \tag{6.219}$$
$$C_{24} = \frac{\varphi}{\delta}, \quad C_{23} = \frac{1}{\delta}$$

6.8.6 Formalisme de Mathis & Zahn (2004)

Les variables

On allège les notations originales:

$$\Phi \Rightarrow \phi,\ \hat{\phi}_2 \Rightarrow \Phi,\ \Psi_2 \Rightarrow \Psi,\ U_2 \Rightarrow U,\ \Lambda_2 \Rightarrow \Lambda,\ \mathcal{A}_2 \Rightarrow A. \tag{6.220}$$

Les indications entre parenthèses, telles que (Equ. B4), renvoient aux équations de l'article de référence.

On note les variables principales par des Majuscules:

1. $x_i,\ i = 1, \ldots, n_c$: abondances par mole
2. $y_1 \equiv \Omega$: vitesse angulaire s^{-1},
3. $y_2 \equiv U$: vitesse de circulation méridienne,
4. $y_3 \equiv \Psi \equiv \frac{\tilde{T}}{T} = \frac{\varphi\Lambda - \Theta}{\delta}$: taux de fluctuation de la température le long d'une isobare
5. $y_4 \equiv \Lambda = \frac{\tilde{\mu}}{\mu}$: taux de fluctuation du poids moléculaire moyen le long d'une isobare
6. $y_5 \equiv \mathcal{T}$: taux de transfert du moment cinétique par unité de masse.
7. $y_6 \equiv \Upsilon$: second membre de l'équation de la vitesse de circulation méridienne.
8. $y_7 \equiv \Phi$: taux de fluctuation du potentiel gravitationnel le long d'une isobare.
9. $y_8 \equiv \Pi$: second membre de l'équation de Poisson.

On ramène le système des équations à un système d'équations du premier ordre totalement implicite. Il est résolu par la méthode de collocation.
Les facteurs numériques sont notés par F_n pour les différencier de ceux du formalisme de Talon et al. (1997).

Transport du moment cinétique

Dans une zone radiative, le moment cinétique par unité de masse vérifie l'équation de transport vertical (Mathis & Zahn (2004), équation 19):

$$\rho\frac{DR^2\Omega}{Dt} = \frac{1}{R^2}\frac{\partial}{\partial R}\left[R^4\rho\left(\frac{\Omega U}{5} + D_{\mathrm{v}}\frac{\partial\Omega}{\partial R}\right)\right] - \rho\dot{\mathcal{M}}_\Omega \Rightarrow \tag{6.221}$$
$$R_\odot^2\frac{Dr^2\Omega}{Dt} = \frac{8\pi R_\odot^6 r^2\rho}{3M_\odot R_\odot^2 r^2\rho\sqrt{\nu}}\frac{\partial}{\partial\nu}\left[r^4\rho\left(\frac{\Omega U}{5} + \frac{8\pi R_\odot^2 r^2\rho D_{\mathrm{v}}}{3M_\odot\sqrt{\nu}}\frac{\partial\Omega}{\partial\nu}\right)\right] - \dot{\mathcal{M}}_\Omega \Rightarrow$$
$$\frac{Dr^2\Omega}{Dt} = \frac{8\pi R_\odot^2}{3M_\odot\sqrt{\nu}}\frac{\partial}{\partial\nu}\left[r^4\rho\left(\frac{\Omega U}{5} + \frac{8\pi R_\odot^2 r^2\rho}{3M_\odot\sqrt{\nu}}D_{\mathrm{v}}\frac{\partial\Omega}{\partial\nu}\right)\right] - \frac{\dot{\mathcal{M}}_\Omega}{R_\odot^2} \Rightarrow$$
$$\sqrt{\nu}\frac{Dr^2\Omega}{Dt} = \frac{\partial}{\partial\nu}\left(\frac{8\pi R_\odot^2}{15M_\odot}r^4\rho\Omega U + \frac{64\pi^2 R_\odot^4}{9M_\odot^2}\frac{r^6\rho^2 D_{\mathrm{v}}}{\sqrt{\nu}}\frac{\partial\Omega}{\partial\nu}\right) - \frac{\dot{\mathcal{M}}_\Omega\sqrt{\nu}}{R_\odot^2}$$

Avec les notations des algorithmes, les équations s'écrivent:

$$F_{26}\Omega - F_{30} = \frac{\partial \mathcal{T}}{\partial \nu}, \quad \mathcal{T} = F_{31}\Omega U + F_{32}\frac{\partial \Omega}{\partial \nu} \tag{6.222}$$
$$F_{26} = \frac{r^2\sqrt{\nu}}{\Delta t},\ F_{30} = (F_{12}^* + F_{13}^*)\sqrt{\nu},\ F_{12}^* = \frac{r^2(t)\Omega(t)}{\Delta t},\ F_{13}^* = -\frac{\dot{\mathcal{M}}_\Omega}{R_\odot^2}$$
$$F_{31} = \frac{8\pi R_\odot^2}{15 M_\odot} r^4 \rho,\ F_{32} = \frac{64\pi^2 R_\odot^4}{9 M_\odot^2}\frac{r^6\rho^2 D_{\rm v}}{\sqrt{\nu}}$$

Vitesse de circulation méridienne U (Equ. B4)

$$0 = \frac{Mg\rho c_{\rm P} T}{LP}\left(\nabla_{\rm ad} - \nabla\right) U \tag{6.223}$$
$$+2\left\{1 - \frac{1}{6\pi G\rho R^2}\frac{\partial R^3\Omega^2}{\partial R} - \frac{\epsilon + \epsilon_g}{\epsilon_m}\right\}\left\{\frac{R^2\Omega^2}{3g^2}\frac{{\rm d}g}{{\rm d}R} - \frac{2R\Omega^2}{3g} - \frac{\rm d}{{\rm d}R}\left(\frac{\Phi}{g}\right)\right\} \tag{6.224}$$
$$+\frac{R\Omega}{3\pi G\rho}\frac{\partial \Omega}{\partial R} \tag{6.225}$$
$$-\frac{\rho_m}{\rho}\left\{\frac{R}{3}\frac{\partial A}{\partial R} - \frac{2H_{\rm T}}{R}\left(1 + \frac{D_h}{K}\right)\Psi\right\} \tag{6.226}$$
$$-\frac{\epsilon + \epsilon_g}{\epsilon_m}\left\{A + \left[f_\epsilon\left(\epsilon_{\rm T} - \delta\right) + \delta\right]\Psi + \left[f_\epsilon\left(\epsilon_\mu + \varphi\right) - \varphi\right]\Lambda\right\} \tag{6.227}$$
$$+\frac{Mc_{\rm P}T}{L}\left(\frac{D\Psi}{Dt} + \phi\frac{D\ln\mu}{Dt}\Lambda\right) \tag{6.228}$$

On développe successivemnt chaque ligne.

Expressions dérivées de Λ (Equ. B5) Pour profiter de la réduction d'un ordre de dérivation (donc d'une variable principale) résultant de l'intégration par parties, comme il a été fait avec le groupe d'équations de Talon/Zahn, on divise les équations (Eq. B4) par ν et on remplace $\mathcal{A}$ par son expression (Eq. B5).

$$A = H_{\rm T}\frac{\partial \Psi}{\partial R} - \left(1 - \delta + \chi_{\rm T}\right)\Psi - \left(\varphi + \chi_\mu\right)\Lambda$$
$$A = \frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial \Psi}{\partial \nu} - \left(1 - \delta + \chi_{\rm T}\right)\Psi - \left(\varphi + \chi_\mu\right)\Lambda$$
$$\frac{2}{3}\frac{\partial A}{\partial \nu} = \frac{\partial}{\partial \nu}\left(\frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}}\frac{\partial \Psi}{\partial \nu}\right) - \frac{\partial}{\partial \nu}\left(\frac{2}{3}\left(1 - \delta + \chi_{\rm T}\right)\Psi\right) - \frac{\partial}{\partial \nu}\left(\frac{2}{3}\left(\varphi + \chi_\mu\right)\Lambda\right) =$$
$$\frac{\partial}{\partial \nu}\left(F_{17}\frac{\partial \Psi}{\partial \nu}\right) + \frac{\partial F_{18}\Psi}{\partial \nu} + \frac{\partial F_{11}\Lambda}{\partial \nu},$$
$$F_{17} = \frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_{\rm T}}{\sqrt{\nu}},\ F_{18} = -\frac{2}{3}(1 - \delta + \chi_{\rm T}),\ F_{11} = -\frac{2}{3}(\varphi + \chi_\mu),$$
$$\frac{1}{\nu}\frac{\partial \ln l}{\partial \ln m}A = \frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial \ln l}{\partial \ln m}\frac{\partial \Psi}{\partial \nu} - \frac{1 - \delta + \chi_{\rm T}}{\nu}\frac{\partial \ln l}{\partial \ln m}\Psi - \frac{\varphi + \chi_\mu}{\nu}\frac{\partial \ln l}{\partial \ln m}\Lambda \rightarrow$$
$$\frac{1}{\nu}\frac{\partial \ln l}{\partial \ln m}A = -F_{12}\frac{\partial \Psi}{\partial \nu} - \frac{1 - \delta + \chi_{\rm T}}{\nu}\frac{\partial \ln l}{\partial \ln m}\Psi - \frac{\varphi + \chi_\mu}{\nu}\frac{\partial \ln l}{\partial \ln m}\Lambda,$$
$$F_{12} = -\frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho H_{\rm T}}{\nu^{\frac{3}{2}}}\frac{\partial \ln l}{\partial \ln m}.$$

(6.223)

$$\frac{Mg\rho c_{\mathrm{P}}T}{LP}\left(\nabla_{\mathrm{ad}}-\nabla\right)U=\nu\left\{\frac{M_{\odot}}{L_{\odot}}\frac{c_{\mathrm{P}}T\sqrt{\nu}}{lH_{\mathrm{P}}}\left(\nabla_{\mathrm{ad}}-\nabla\right)U\right\} \tag{6.229}$$

(6.230)

(6.224)

$$\frac{2}{\nu}\left\{1-\frac{1}{6\pi G\rho R^2}\frac{\partial R^3\Omega^2}{\partial R}-\frac{\epsilon+\epsilon_g}{\epsilon_m}\right\}=\frac{2}{\nu}-\frac{1}{3\pi G\rho R^2\nu}\left(3R^2\Omega^2+2R^3\Omega\frac{\partial\Omega}{\partial R}\right)-\frac{2}{\nu}\frac{\partial\ln l}{\partial\ln m}=$$
$$\frac{2}{\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)-\frac{\Omega^2}{\pi G\rho\nu}-\frac{2R}{3\pi G\rho\nu}\Omega\frac{\partial\Omega}{\partial R}$$
$$\frac{R^2}{3g^2}\frac{\mathrm{d}g}{\mathrm{d}R}\Omega^2-\frac{2R\Omega^2}{3g}-\frac{\mathrm{d}}{\mathrm{d}R}\left(\frac{\Phi}{g}\right)=\left(\frac{R}{3g}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-\frac{2R}{3g}\right)\Omega^2+\frac{1}{Rg}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Phi-\frac{1}{g}\frac{\partial\Phi}{\partial R}=$$
$$\frac{R}{3g}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^2+\frac{1}{gR}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Phi-\frac{1}{g}\frac{\partial\Phi}{\partial R}$$
$$2\{-\}\{-\}=\nu\left(\frac{2}{\nu}\{-\}\{-\}\right)=$$
$$\nu\left\{\frac{2R}{3g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^2+\frac{2}{g\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\Phi\right.$$
$$-\frac{2}{g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\frac{\partial\Phi}{\partial R}-\frac{R}{3\pi Gg\rho\nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^4-\frac{1}{\pi Gg\rho\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega^2\Phi$$
$$+\frac{1}{\pi Gg\rho\nu}\Omega^2\frac{\partial\Phi}{\partial R}-\frac{2R^2}{9\pi Gg\rho\nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^3\frac{\partial\Omega}{\partial R}$$
$$\left.-\frac{2}{3\pi Gg\rho\nu}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega\Phi\frac{\partial\Omega}{\partial R}+\frac{2R}{3\pi Gg\rho\nu}\Omega\frac{\partial\Omega}{\partial R}\frac{\partial\Phi}{\partial R}\right\}$$

(6.225)

$$\frac{R\Omega}{3\pi G\rho}\frac{\partial\Omega}{\partial R}=\nu\left\{\frac{8R_{\odot}^3}{9M_{\odot}G}\frac{r^3}{\nu^{\frac{3}{2}}}\Omega\frac{\partial\Omega}{\partial\nu}\right\}$$

(6.226)

$$\frac{\rho_m}{\rho}\left\{\frac{R}{3}\frac{\partial A}{\partial R}-\frac{2H_{\mathrm{T}}}{R}\left(1+\frac{D_h}{K}\right)\Psi\right\}=$$
$$\frac{3M_{\odot}}{4\pi R_{\odot}^3}\frac{\nu^{\frac{3}{2}}}{r^3\rho}\frac{R_{\odot}r}{3}\frac{8\pi R_{\odot}^2}{3M_{\odot}}\frac{r^2\rho}{\sqrt{\nu}}\frac{\partial A}{\partial\nu}-\frac{3M_{\odot}}{4\pi R_{\odot}^3}\frac{\nu^{\frac{3}{2}}}{r^3\rho}\frac{2H_{\mathrm{T}}}{R_{\odot}r}\left(1+\frac{D_h}{K}\right)\Psi=$$
$$\nu\frac{2}{3}\frac{\partial A}{\partial\nu}-\frac{3M_{\odot}}{2\pi R_{\odot}^4}\frac{H_{\mathrm{T}}\nu^{\frac{3}{2}}}{r^4\rho}\left(1+\frac{D_h}{K}\right)\Psi=$$
$$\nu\left\{\frac{\partial}{\partial\nu}\left(F_{17}\frac{\partial\Psi}{\partial\nu}\right)+\frac{\partial F_{18}\Psi}{\partial\nu}+\frac{\partial F_{11}\Lambda}{\partial\nu}-\frac{3M_{\odot}}{2\pi R_{\odot}^4}\frac{H_{\mathrm{T}}\sqrt{\nu}}{r^4\rho}\left(1+\frac{D_h}{K}\right)\Psi\right\}$$

(6.227)

$$\begin{aligned}
&\frac{\epsilon+\epsilon_g}{\epsilon_m}\left\{A+\left[f_\epsilon\left(\epsilon_{\mathrm{T}}-\delta\right)+\delta\right]\Psi+\left[f_\epsilon\left(\epsilon_\mu+\varphi\right)-\varphi\right]\Lambda\right\}=\\
&\nu\left\{\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}A+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left[\left(f_\epsilon\left(\epsilon_{\mathrm{T}}-\delta\right)+\delta\right)\right]\Psi+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left[\left(f_\epsilon\left(\epsilon_\mu+\varphi\right)-\varphi\right)\right]\Lambda\right\}=\\
&\nu\left\{-F_{12}\frac{\partial\Psi}{\partial\nu}+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(f_\epsilon\left(\epsilon_{\mathrm{T}}-\delta\right)+2\delta-1-\chi_{\mathrm{T}}\right)\Psi+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(f_\epsilon\left(\epsilon_\mu+\varphi\right)-2\varphi-\chi_\mu\right)\Lambda\right\}=\\
&=\nu\left\{-F_{12}\frac{\partial\Psi}{\partial\nu}+\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_{\mathrm{T}}-\delta\right)+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\delta-1-\chi_{\mathrm{T}}\right)\right)\Psi\right.\\
&\left.+\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_\mu+\varphi\right)-\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\varphi+\chi_\mu\right)\right)\Lambda\right\}
\end{aligned}$$

(6.228)

$$\begin{aligned}
&\frac{Mc_{\mathrm{P}}T}{L}\left(\frac{D\Psi}{Dt}+\phi\frac{D\ln\mu}{Dt}\Lambda\right)=\\
&\nu\left\{\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T}{l\Delta t}\Psi+\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\phi}{l\Delta t}\left(\ln\mu-\ln\mu^t\right)\Lambda-\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\Psi^t}{l\Delta t}\right\}
\end{aligned}$$

Regroupement

$$\begin{aligned}
0=&\frac{M_\odot}{L_\odot}\frac{c_{\mathrm{P}}T\sqrt{\nu}}{lH_{\mathrm{P}}}\left(\nabla_{\mathrm{ad}}-\nabla\right)U\\
&+\frac{2R}{3g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^2+\frac{2}{g\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\Phi\\
&-\frac{2}{g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\frac{\partial\Phi}{\partial R}-\frac{R}{3\pi Gg\rho\nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^4-\frac{1}{\pi Gg\rho\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega^2\Phi\\
&+\frac{1}{\pi Gg\rho\nu}\Omega^2\frac{\partial\Phi}{\partial R}-\frac{2R^2}{9\pi Gg\rho\nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^3\frac{\partial\Omega}{\partial R}\\
&-\frac{2}{3\pi Gg\rho\nu}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega\Phi\frac{\partial\Omega}{\partial R}+\frac{2R}{3\pi Gg\rho\nu}\Omega\frac{\partial\Omega}{\partial R}\frac{\partial\Phi}{\partial R}+\frac{8R_\odot^3}{9M_\odot G}\frac{r^3}{\nu^{\frac{3}{2}}}\Omega\frac{\partial\Omega}{\partial\nu}\\
&-\frac{\partial}{\partial\nu}\left(F_{17}\frac{\partial\Psi}{\partial\nu}\right)-\frac{\partial F_{18}\Psi}{\partial\nu}-\frac{\partial F_{11}\Lambda}{\partial\nu}+\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\mathrm{T}}\sqrt{\nu}}{r^4\rho}\left(1+\frac{D_h}{K}\right)\Psi\\
&+F_{12}\frac{\partial\Psi}{\partial\nu}-\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_{\mathrm{T}}-\delta\right)+\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\delta-1-\chi_{\mathrm{T}}\right)\right)\Psi\\
&-\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_\mu+\varphi\right)-\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\varphi+\chi_\mu\right)\right)\Lambda\\
&+\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T}{l\Delta t}\Psi+\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\phi}{l\Delta t}\left(\ln\mu-\ln\mu^t\right)\Lambda-\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\Psi^t}{l\Delta t}
\end{aligned}$$

Ou encore:

$$
\begin{aligned}
0 = & -\frac{R}{3\pi G g \rho \nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^4 - \frac{2R^2}{9\pi G g \rho \nu}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^3\frac{\partial\Omega}{\partial R} \\
& +\frac{2R}{3g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)\Omega^2 + \frac{8R_\odot^3}{9M_\odot G}\frac{r^3}{\nu^{\frac{3}{2}}}\Omega\frac{\partial\Omega}{\partial\nu} + \frac{1}{\pi G g\rho\nu}\Omega^2\frac{\partial\Phi}{\partial R} \\
& -\frac{1}{\pi G g\rho\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega^2\Phi + \frac{2R}{3\pi G g\rho\nu}\Omega\frac{\partial\Omega}{\partial R}\frac{\partial\Phi}{\partial R} - \frac{2}{3\pi G g\rho\nu}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\Omega\Phi\frac{\partial\Omega}{\partial R} \\
& +\frac{2}{g\nu R}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\Phi \\
& -\frac{2}{g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\frac{\partial\Phi}{\partial R} - \frac{\partial F_{18}\Psi}{\partial\nu} - \frac{\partial F_{11}\Lambda}{\partial\nu} - \frac{\partial}{\partial\nu}\left(F_{17}\frac{\partial\Psi}{\partial\nu}\right) + F_{12}\frac{\partial\Psi}{\partial\nu} \\
& -\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_{\mathrm{T}}-\delta\right) + \frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\delta-1-\chi_{\mathrm{T}}\right) - \frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T}{l\Delta t}\right. \\
& \left.-\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_{\mathrm{T}}\sqrt{\nu}}{r^4\rho}\left(1+\frac{D_h}{K}\right)\right)\Psi \\
& -\left(\frac{M_\odot}{L_\odot}\frac{\varepsilon_{\mathrm{nuc}}\sqrt{\nu}}{l}\left(\epsilon_\mu+\varphi\right) - \frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\varphi+\chi_\mu\right) - \frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\phi}{l\Delta t}\left(\ln\mu-\ln\mu^t\right)\right)\Lambda \\
& -\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_{\mathrm{P}}T\Psi^t}{l\Delta t} + \frac{M_\odot}{L_\odot}\frac{c_{\mathrm{P}}T\sqrt{\nu}}{lH_{\mathrm{P}}}\left(\nabla_{\mathrm{ad}}-\nabla\right)U
\end{aligned}
$$

La formulation utilisée s'écrit:

$$
\begin{aligned}
& F_1\Omega^4 + F_2\Omega^3\frac{\partial\Omega}{\partial\nu} + F_3\Omega^2 + F_4\Omega\frac{\partial\Omega}{\partial\nu} + F_5\Omega^2\frac{\partial\Phi}{\partial\nu} + F_6\Omega^2\Phi + F_7\Omega\frac{\partial\Omega}{\partial\nu}\frac{\partial\Phi}{\partial\nu} \\
& +F_8\Omega\Phi\frac{\partial\Omega}{\partial\nu} + F_9\Phi + F_{10}\frac{\partial\Phi}{\partial\nu} - \frac{\partial F_{18}\Psi}{\partial\nu} - \frac{\partial F_{11}\Lambda}{\partial\nu} - \frac{\partial}{\partial\nu}\left(F_{17}\frac{\partial\Psi}{\partial\nu}\right) + F_{12}\frac{\partial\Psi}{\partial\nu} \\
& +F_{13}\Psi + F_{14}\Lambda + F_{15} + F_{16}U = 0 \Longrightarrow \\
& (F_1\Omega^2+F_3)\Omega^2 + (F_2\Omega^2+F_8\Phi+F_4)\Omega\frac{\partial\Omega}{\partial\nu} + (F_6\Omega^2+F_9)\Phi + F_7\Omega\frac{\partial\Omega}{\partial\nu}\frac{\partial\Phi}{\partial\nu} \\
& +\left(F_5\Omega^2+F_{10}\right)\frac{\partial\Phi}{\partial\nu} + F_{12}\frac{\partial\Psi}{\partial\nu} + F_{13}\Psi + F_{14}\Lambda + F_{15} + F_{16}U = \frac{\partial\Upsilon}{\partial\nu} \qquad (6.231) \\
& \Upsilon = F_{17}\frac{\partial\Psi}{\partial\nu} + F_{11}\Lambda + F_{18}\Psi \qquad (6.232)
\end{aligned}
$$

Les coefficients ont pour expressions:

$$F_1\Omega^4,\ F_1 = -\frac{R_\odot}{3\pi G}\frac{r}{\rho\nu g}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right),\ F_2\Omega^3\frac{\partial\Omega}{\partial\nu},\ F_2 = -\frac{16R_\odot^4}{27M_\odot G}\frac{r^4}{g\nu^{\frac{3}{2}}}\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right)$$

$$F_3\Omega^2, F_3 = \frac{2R_\odot}{3}\frac{r}{g\nu}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)\left(\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}-2\right),\ F_4\Omega\frac{\partial\Omega}{\partial\nu},\ F_4 = \frac{8R_\odot^3}{9M_\odot G}\frac{r^3}{\nu^{\frac{3}{2}}}$$

$$F_5\Omega^2\frac{\partial\Phi}{\partial\nu},\ F_5 = \frac{8R_\odot^2}{3M_\odot G}\frac{r^2}{g\nu^{\frac{3}{2}}},\ F_6\Omega^2\Phi, F_6 = -\frac{1}{\pi GR_\odot}\frac{1}{g\rho\nu r}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R},\ F_7\Omega\frac{\partial\Omega}{\partial\nu}\frac{\partial\Phi}{\partial\nu},\ F_7 = \frac{128\pi R_\odot^5}{27M_\odot^2 G}\frac{r^5\rho}{g\nu^2}$$

$$F_8\Omega\Phi\frac{\partial\Omega}{\partial\nu},\ F_8 = -\frac{16R_\odot^2}{9M_\odot G}\frac{r^2}{g\nu^{\frac{3}{2}}}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R},\ F_9\Phi,\ F_9 = \frac{2}{R_\odot}\frac{1}{g\nu r}\frac{\mathrm{d}\ln g}{\mathrm{d}\ln R}\left(1-\frac{\partial\ln l}{\partial\ln m}\right)$$

$$F_{10}\frac{\partial\Phi}{\partial\nu}, F_{10} = -\frac{16\pi R_\odot^2}{3M_\odot}\frac{r^2\rho}{g\nu^{\frac{3}{2}}}\left(1-\frac{\partial\ln l}{\partial\ln m}\right),\ \frac{\partial F_{11}\Lambda}{\partial\nu}, F_{11} = -\frac{2}{3}(\varphi+\chi_\mu)$$

$$F_{12}\frac{\partial\Psi}{\partial\nu},\ F_{12} = -\frac{8\pi R_\odot^2}{3M_\odot}\frac{r^2\rho H_\mathrm{T}}{\nu^{\frac{3}{2}}}\frac{\partial\ln l}{\partial\ln m}$$

$$F_{13}\Psi,\ F_{13} = -\left(F_1^*+F_2^*+F_3^*+F_4^*\right),\ F_1^* = \frac{M_\odot}{L_\odot}\frac{\varepsilon_\mathrm{nuc}\sqrt{\nu}}{l}\left(\epsilon_\mathrm{T}-\delta\right),$$

$$F_2^* = \frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\delta-1-\chi_\mathrm{T}\right),\ F_3^* = -\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_\mathrm{P}T}{l\Delta t},\ F_4^* = -\frac{3M_\odot}{2\pi R_\odot^4}\frac{H_\mathrm{T}\sqrt{\nu}}{r^4\rho}\left(1+\frac{D_h}{K}\right)$$

$$F_{14}\Lambda,\ F_{14} = -\left(F_5^*+F_6^*+F_7^*\right),\ F_5^* = \frac{M_\odot}{L_\odot}\frac{\varepsilon_\mathrm{nuc}\sqrt{\nu}}{l}\left(\epsilon_\mu+\varphi\right),$$

$$F_6^* = -\frac{1}{\nu}\frac{\partial\ln l}{\partial\ln m}\left(2\varphi+\chi_\mu\right),\ F_7^* = -\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_\mathrm{P}T\phi}{l\Delta t}\left(\ln\mu-\ln\mu^t\right)$$

$$F_{15},\ F_{15} = -\frac{M_\odot}{L_\odot}\frac{\sqrt{\nu}c_\mathrm{P}T\Psi^t}{l\Delta t},\ F_{16}U,\ F_{16} = \frac{M_\odot}{L_\odot}\frac{c_\mathrm{P}T\sqrt{\nu}}{lH_\mathrm{P}}\left(\nabla_\mathrm{ad}-\nabla\right)$$

$$\frac{\partial}{\partial\nu}\left(F_{17}\frac{\partial\Psi}{\partial\nu}\right),\ F_{17} = \frac{16\pi R_\odot^2}{9M_\odot}\frac{r^2\rho H_\mathrm{T}}{\sqrt{\nu}},\ F_{18}\Psi,\ F_{18} = -\frac{2}{3}(1-\delta+\chi_\mathrm{T})$$

Relation barocline (Equ. B6)

$$\Theta = \frac{R^2}{3g}\frac{\partial\Omega^2}{\partial R} = \frac{2R^2}{3g}\Omega\frac{\partial\Omega}{\partial R} = \varphi\Lambda-\delta\Psi \Longrightarrow \frac{16\pi R_\odot^4}{9M_\odot}\frac{r^4\rho}{g\sqrt{\nu}}\Omega\frac{\partial\Omega}{\partial\nu}-\varphi\Lambda+\delta\Psi = 0 \Longrightarrow$$
$$F_{19}\Omega\frac{\partial\Omega}{\partial\nu}+F_{20}\Lambda+F_{21}\Psi = 0 \tag{6.233}$$
$$F_{19} = \frac{16\pi R_\odot^4}{9M_\odot}\frac{r^4\rho}{g\sqrt{\nu}},\ F_{20} = -\varphi,\ F_{21} = \delta$$

Fluctuations du poids moléculaire (Equ. B7)

$$\frac{D\Lambda}{Dt}+\left(\frac{6D_\mathrm{h}}{R^2}-\frac{D\ln\mu}{Dt}\right)\Lambda-\frac{U\nabla_\mu}{H_\mathrm{P}} = 0 \Longrightarrow\ F_{22}U+F_{23}\Lambda+F_{24} = 0 \tag{6.234}$$
$$F_{22} = -\frac{\nabla_\mu}{H_\mathrm{P}},\ F_{23} = F_8^*+F_9^*,\ F_8^* = \frac{6}{R_\odot^2}\frac{D_\mathrm{h}}{r^2},\ F_9^* = -\frac{\ln\mu-\ln\mu^t-1}{\Delta t},\ F_{24} = -\frac{\Lambda^t}{\Delta t}$$

Pour la formation des algorithmes, cette expression a été substituée à Λ dans les équations précédentes.

Equation de Poisson (Equ. B8)

$$0 = \frac{1}{R}\frac{\partial^2 R\Phi}{\partial R^2} - \left(\frac{6}{R^2} + \frac{4\pi G}{g}\frac{\partial \rho}{\partial R}\right)\Phi + \frac{8\pi G}{3}\frac{\rho R}{g}\Omega^2 - \frac{4\pi G}{3g}\frac{\partial}{\partial R}\left(\rho R^2\Omega^2\right)$$

$$\frac{1}{R}\frac{\partial^2 R\Phi}{\partial R^2} = \frac{1}{R}\frac{\partial}{\partial R}\left(\Phi + R\frac{\partial \Phi}{\partial R}\right)$$

$$\frac{8\pi G}{3}\frac{\rho R}{g}\Omega^2 - \frac{4\pi G}{3g}\frac{\partial}{\partial R}\left(\rho R^2\Omega^2\right) =$$

$$\frac{4\pi G}{3g}2R\rho\Omega^2 - \frac{4\pi G}{3g}\left(2R\rho + R^2\frac{\partial \rho}{\partial R}\right)\Omega^2 - \frac{8\pi G}{3g}\rho R^2\Omega\frac{\partial \Omega}{\partial R} =$$

$$-\frac{4\pi G}{3}\frac{R^2\rho}{g}\frac{\partial \ln \rho}{\partial R}\Omega^2 - \frac{8\pi G}{3}\frac{\rho R^2}{g}\Omega\frac{\partial \Omega}{\partial R} \Longrightarrow$$

$$0 = \frac{8\pi R_\odot}{3M_\odot}\frac{r\rho}{\sqrt{\nu}}\frac{\partial}{\partial \nu}\left(\Phi + \frac{8\pi R_\odot^3}{3M_\odot}\frac{r^3\rho}{\sqrt{\nu}}\frac{\partial \Phi}{\partial \nu}\right) - \left(\frac{6}{R_\odot^2 r^2} + \frac{32\pi^2 R_\odot^2 G}{3M_\odot}\frac{r^2\rho^2}{g\sqrt{\nu}}\frac{\partial \ln \rho}{\partial \nu}\right)\Phi$$

$$-\frac{32\pi^2 R_\odot^4 G}{9M_\odot}\frac{r^4\rho^2}{g\sqrt{\nu}}\frac{\partial \ln \rho}{\partial \nu}\Omega^2 - \frac{64\pi^2 R_\odot^4 G}{9M_\odot}\frac{r^4\rho^2}{g\sqrt{\nu}}\Omega\frac{\partial \Omega}{\partial \nu}$$

En multipliant par $\frac{3M_\odot}{8\pi R_\odot}\frac{\sqrt{\nu}}{r\rho}$:

$$\frac{\partial}{\partial \nu}\left(\Phi + \frac{8\pi R_\odot^3}{3M_\odot}\frac{r^3\rho}{\sqrt{\nu}}\frac{\partial \Phi}{\partial \nu}\right) - \left(\frac{9M_\odot}{4\pi R_\odot^3}\frac{\sqrt{\nu}}{r^3\rho} + 4\pi R_\odot G\frac{r\rho}{g}\frac{\partial \ln \rho}{\partial \nu}\right)\Phi$$

$$-\frac{4\pi R_\odot^3 G}{3}\frac{r^3\rho}{g}\frac{\partial \ln \rho}{\partial \nu}\Omega^2 - \frac{8\pi R_\odot^3 G}{3}\frac{r^3\rho}{g}\Omega\frac{\partial \Omega}{\partial \nu} = 0$$

L'équation de Poisson se ramène à:

$$F_{29}\Omega^2 + F_{28}\Omega\frac{\partial \Omega}{\partial \nu} + F_{27}\Phi = \frac{\partial \Pi}{\partial \nu} \tag{6.235}$$

$$\Pi = \Phi + F_{25}\frac{\partial \Phi}{\partial \nu} \tag{6.236}$$

$$F_{25} = \frac{8\pi R_\odot^3}{3M_\odot}\frac{r^3\rho}{\sqrt{\nu}},\ F_{27} = F_{10}^* + F_{11}^*,\ F_{10}^* = \frac{9M_\odot}{4\pi R_\odot^3}\frac{\sqrt{\nu}}{r^3\rho},\ F_{11}^* = 4\pi G R_\odot\frac{r\rho}{g}\frac{\partial \ln \rho}{\partial \nu},$$

$$F_{28} = \frac{8\pi G R_\odot^3}{3}\frac{r^3\rho}{g},\ F_{29} = \frac{4\pi R_\odot^3 G}{3}\frac{r^3\rho}{g}\frac{\partial \ln \rho}{\partial \nu}$$

6.8.7 Les conditions physiques dans les zones mélangées

Les zones mélangées, i.e. zones convectives éventuellement overshootées, apparaissent, disparaissent au cours de l'évolution. L'observation semble montrer que la vitesse angulaire y est constante, il y a ***rotation solide***. Dans le milieu turbulent la vitesse de circulation méridienne U n'a pas de sens physique. De fait, le gradient de la vitesse angulaire n'est pas exactement nul, il correspond à un transfert de moment cinétique qui assure la quasi uniformité de la vitesse angulaire. Ce modèle correspond bien à l'image que l'on peut se faire d'un mélange qui, bien que rapide, ne peut être instantanné. Son échelle de temps est toutefois petite en regard des autres échelles de temps en présence, celles de Kelving-Helmotz en particulier.

Avec une rotation solide $\frac{\partial \Omega}{\partial \nu} = 0$. Le moment cinétique par unité de masse $R^2\Omega$ n'est pas constant. Le moment cinétique total dans une zone convective varie au cours de l'évolution, à cause d'une part des variations des condions physiques, température, densité composition chimique et, d'autre part en raison des modifications de son étendue.A l'image de ce qui est fait pour la composition chimique, Cesam2k20 propose deux formalismes pour assurer l'uniformisation de la vitesse angulaire.

Formalisme par diffusion : L'uniformisation de la vitesse angulaire dans les zones convectives est réalisée par une diffusion verticale de coefficient très grand devant l'unité, $D_{\rm v} >> 1$. Les équations de transport vertical du moment cinétique 6.210 (Page 103) et 6.222 (Page 108), dans les zone convective, deviennent respectivement:

$$C_1\Omega - C_2 = \frac{\partial \mathcal{T}}{\partial \nu}, \quad \mathcal{T} = C_4 \frac{\partial \Omega}{\partial \nu} \tag{6.237}$$

$$F_{26}\Omega - F_{30} = \frac{\partial \mathcal{T}}{\partial \nu}, \quad \mathcal{T} = F_{32} \frac{\partial \Omega}{\partial \nu} \tag{6.238}$$

Dans l'équation 6.234 (Page 112) d'évolution des fluctuations du poids moléculaire, $D_{\rm h} >> 1$ entraîne $\Lambda = 0$ et, de là, $\Psi = 0$ (*cf.* (6.233)). Ces conditions entraînent la discontinuité des gradients des fonctions inconnues aux limites entre zones radiatives et zones mélangées. Pour éviter d'introduire des limites internes mobiles et en nombre variable, ce qui compliquerai l'intégration numérique, des bases de B-splines continues non dérivables sont utilisées pour la diffusion du moment cinétique. Les discontinuités des gradients sont alors implicitement prises en compte par Cesam2k20.

Ainsi dans les zones mélangées:

$$\frac{\partial \Omega}{\partial \nu} \sim 0,\ U = 0,\ \Theta = 0,\ \Lambda = 0,\ \Psi = 0,\ \Upsilon = 0.$$

Dans une zone convective, le taux de transport vertical du moment cinétique se réduit à :

$$\mathcal{T} = \frac{64\pi^2 R_\odot^4}{9 M_\odot^2} \frac{r^6 \rho^2 D_{\rm v}}{\sqrt{\nu}} \frac{\partial \Omega}{\partial \nu}$$

et l'équation de transport vertical du moment cinétique devient:

$$C_1\Omega - C_2 = \frac{\partial \mathcal{T}}{\partial \nu}, \quad \mathcal{T} = C_4 \frac{\partial \Omega}{\partial \nu}$$

$$F_{26}\Omega - F_{30} = \frac{\partial \mathcal{T}}{\partial \nu}, \quad \mathcal{T} = F_{32} \frac{\partial \Omega}{\partial \nu}$$

Dans une zone radiative, la diffusion verticale du moment cinétique procède de deux processus, une advection liée à la vitesse de circulation méridienne et une diffusion proprement dite. Le processus d'échange de moment cinétique entre une zone convectiveet une zone radiativeadjacentes est inconnu. On admet que la vitesse angulaire et le taux de transfert vertical du moment cinétique sont des fonctions continues sur les limites zone convective/zone radiative. En raison des mouvements turbulents à grande échelle dont la zone convective est le siège, le taux de transfert vertical de moment cinétique entre une zone convective et une zone radiative adjacentes est supposé correspondre au taux du processus d'advection affectant la zone radiative.

$$\mathcal{T} = C_3 U \Omega, \quad \mathcal{T} = F_{31} U \Omega$$

La résolution de ces équations conduit à la connaissance de la valeur de la vitesse de circulation méridienne coté radiatif des limites RZ/CZ.

La vitesse de circulation méridienne U n'intervient pas dans les équations vérifiées dans les zone convective, *cf.* équation 6.237 (Page 114). Du point de vue algorithmique, il est possible d'utiliser la variable attachée à U dans l'intégration de l'équation 6.237 (Page 114) et de supposer U comme une variable continue aux limites zone radiative/ zone convective. Ce dispositif, utilisé dans Cesam2k20, simplifie les algorithmes. Dans les zones convectives, on définit une vitesse U par:

$$U = \frac{\mathcal{T}}{C_3 \Omega}, \quad U = \frac{\mathcal{T}}{F_{31}\Omega}$$

6.8.8 Les conditions physiques aux limites

Le problème différentiel, dans son ensemble est constitué d'un problème différentiel de conditions initiales pour Λ, d'une définition pour Ψ, de problèmes problème aux limites de conditions initiales pour Ω, U et Φ. Pour sa résolution numérique, il est ramené à un problème aux limites de conditions initiales contitué, suivant le formalisme retenu, de 7 ou 8 équations différentielles et scalaires. La méthode numérique utilisée nécessite la connaissance d'un nombre égal de conditions limites.

Le centre: ($\nu = \nu_0 = 0$), suivant l'état d'évolution le centre peut être situé dans une zone radiative ou dans une zone convective. Les conditions limites utilisées sont:

$$\frac{\partial\Omega}{\partial\nu} = 0, \quad U = 0, \quad \Lambda = 0, \quad \Upsilon = 0 \quad \text{(formalisme } \texttt{tz97}\text{)}$$
$$\frac{\partial\Omega}{\partial\nu} = 0, \quad U = 0, \quad \Lambda = 0, \quad \Upsilon = 0, \quad \Phi = 0 \quad \text{(formalisme } \texttt{mz04}\text{)}$$

La surface: La partie externe de l'enveloppe, zone d'ionisation de l'hydrogène, est toujours une zone convective.

Limite externe pour l'équation de Poisson: Suivant Mathis & Zahn (2004) le potentiel gravitationnel Φ vérifie (correction faite d'une erreur d'homogénéité):

$$3\Phi + R\frac{\partial\Phi}{\partial R} = 0 \Rightarrow \Phi = -\frac{R}{3}\frac{\partial\Phi}{\partial R} \Rightarrow \Phi = -\frac{F_{25}}{3}\frac{\partial\Phi}{\partial\nu} \Rightarrow$$
$$\Pi = \Phi + F_{25}\frac{\partial\Phi}{\partial\nu} = \frac{2}{3}F_{25}\frac{\partial\Phi}{\partial\nu} = -2\Phi$$

Les conditions limites utilisées sont:

$$U = 0, \quad \Psi = 0, \quad \Theta = 0 \quad \text{(formalisme } \texttt{tz97}\text{)}$$
$$\mathcal{T} = 0, \quad \Psi = 0, \quad \Pi + 2\Phi = 0 \quad \text{(formalisme } \texttt{mz04}\text{)}$$

6.8.9 Les conditions initiales

Des conditions initiales doivent être définies pour la vitesse angulaire Ω_0, la vitesse de circulation méridienne U_0 et le taux de fluctuation du poids moléculaire moyen Λ_0. Les conditions initiales sont établies par la routine PRIVATE `initialise_rota` du module `MOD_CESAM`.

Vitesse angulaire initiale: on ne dispose pas de données observationnelles permettant d'inférer la répartition spatiale de la vitesse angulaire d'un modèle initial de PMS ou de séquence principale d'âge zéro. Faute de mieux, Cesam2k20 suppose la rotation solide. La vitesse angulaire initiale est déduite de la variable `W_ROT` du fichier de données `mon_modele.don`, *cf.* §3.2 (Page 14).

Il est toutefois possible d'utiliser un profil de rotation grossièrement similaire à celui que l'on peut inférer des observations de sismologie solaire:

$$\Omega_0(r,0) = \Omega_0\left\{1 + q\left[1 + \cos\left(\pi\frac{r}{R_\star}\right)\right]\right\}$$

Ω_0 est la vitesse angulaire initiale, éventuellement déduite de la période de rotation et du rayon total initial, *cf.* §3.2 (Page 14). $q \geq -1$ est un paramètre permettant de fixer le rapport entre la vitesse angulaire au centre et à la surface, $q = 0$ correspond à une rotation rigide. q est la variable `w_form` du module `MOD_DONNEES` elle y est fixée à 0. Pour utiliser une valeur différente, il convient de la redéfinir à l'aide d'un fichier de "réglages", *cf.* §3.16 (Page 33). Avec $q \neq 0$, le profil de Ω_0 n'est pas iniformisé dans les zone convective. Ω est initialisé dans la routine `w_initial` du module `MOD_CESAM`.

Vitesse de circulation méridienne initiale: cette quantité n'est pas accessible à l'observation. De façon euristique, dans la routine `u_initial` du module `MOD_CESAM`, U_0 est obtenu par la relation empirique:

$$U_0(R) = \frac{|w_{\rm rot}|u_i}{2-(R/R_\star)^2} \tag{6.239}$$

où u_i est un paramètre fixé à 10^{-4} et $w_{\rm rot}$ la valeur de la variable `W_ROT` du fichier de données `mon_modele.don`.

Taux de fluctuation du poids moléculaire moyen: Λ est supposé nul à l'instant initial: $\Lambda_0 = 0$.

Conditions initiales pour Θ***, et*** Ψ***:*** elles sont nécessaires pour la résolution numérique, la formulation des équations étant implicite. Cesam2k20 les calcule à partir des relations les définissant, *cf.* équation 6.216 (Page 106) et équation 6.219 (Page 107), en utilisant les valeurs initiales retenues pour Ω et Λ.

Conditions initiales pour Φ***:*** Le potentiel gravitationnel initial doit vérifier l'équation de Poisson. Il est obtenu en intégrant l'équation de Poisson du formalisme de Mathis & Zahn (2004) par éléments finis.

6.8.10 Pertes / gains de moment cinétique

Par l'intermédiaire de la routine générique `pertw` Cesam2k20 offre la possibilité d'utiliser différentes formes de pertes/gains de moment cinétique. Chacune d'elles dépend d'un paramètre libre, respectivement $a,\ \gamma,\ \varpi$, transmis par la varible `p_pertw` du fichier de données, *cf.* §3.2 (Page 14). Une valeur négative/positive de ce paramètre correspond à une perte/gain de moment cinétique. Cette perte/gain de moment cinétique ne concerne que la zone convective externe. Les formes suivantes sont implantées dans Cesam2k20:

- `pertw_sch`: La variation temporelle de moment cinétique par unité de masse et de temps, proportionnelle à Ω^3 (Schumanish), est modélisée par:

$$\dot{\mathcal{M}}_\Omega = aR^2\Omega_{\rm s}^3, \quad |a| \sim 1.10^{-9}{\rm s}. \tag{6.240}$$

- `pertw_loc`: Perte / gain de moment cinétique proportionnel à l'énergie cinétique de rotation locale. La variation temporelle de moment cinétique par unité de masse est modélisée par:

$$\dot{\mathcal{M}}_\Omega = \gamma R^2\Omega^2, \quad |\gamma| \sim 10^{-13} \sim 10^{-14}. \tag{6.241}$$

- `pertw_ptm`: Perte / gain de moment cinétique conséquence d'une perte / gain de masse $\dot{\mathcal{M}}$. Dans le cas d'une perte de masse, $\dot{\mathcal{M}} < 0$, on suppose qu'une couche externe se détache et emporte avec elle son moment cinétique, alors $\varpi = 1$. La perte de moment cinétique par unité de masse est approchée par:

$$\dot{\mathcal{M}}_\Omega = \varpi\frac{\dot{\mathcal{M}}}{M_{\rm ZC}}R_{\rm ZC}^2\Omega_{\rm ZC}. \tag{6.242}$$

  Dans le cas d'un gain de masse, $\dot{\mathcal{M}} > 0$, on suppose que l'apport de masse est animé d'une vitesse angulaire égale à celle de la couche externe avec un rayon de giration égal au rayon de l'étoile,$\varpi > 0$ peut être utilisé pour adapter ces dispositions.

- `pertw_0`: On ignore les perte / gain de moment cinétique. La variation temporelle de moment cinétique est modélisée par:

$$\dot{\mathcal{M}}_\Omega = 0. \tag{6.243}$$

Chutes de planètoïdes: un gain/perte de moment cinétique de la zone convective externe résulte des chutes de planètoïdes, *cf.* §6.7.2 (Page 87). En notant $\Omega_{\rm pl}$ la vitesse angulaire des planètoïdes, $R_{\rm pl} \geq R_\star$ le rayon de giration des planètoïdes, l'apport de moment cinétique par unité de masse est:

$$\dot{\mathcal{M}}_\Omega = \frac{N_{\rm P} M_\oplus}{M_{\rm ZC}} \mathcal{P}(t) R_{\rm pl}^2 \Omega_{\rm pl}.$$

$N_{\rm P} > 0$ est le nombre total de planètoïdes de masse terrestre, $M_\oplus$, reçus par l'étoile, $\mathcal{P}(t)$ la fonction décrivant la dépendance temporelle, *cf.* §3.15.5 (Page 33). Une vitesse angulaire positive des planètoïdes accroit le moment cinétique total de la zone convective externe.

6.9 La convection

6.9.1 Critères de convection

L'énergie est transportée par convection lorsqu'est vérifié *cf.* Cox & Giuli (1968, p. 276) ou encore **?**, p. 39:

- soit le critère de Ledoux, qui tient compte du gradient de composition chimique:

$$\nabla_{\rm rad} < \nabla_{\rm ad}^* + \frac{\varphi}{\delta}\nabla_\mu \;\sim\; \frac{4-3\beta}{\beta}\left(\nabla_{\rm ad}^* - \nabla_{\rm rad}\right) + \nabla_\mu < 0, \;\; \left(\varphi \equiv \frac{d\ln\rho}{d\ln \text{ţ}}, \;\; \nabla_\mu \equiv \frac{d\ln \text{ţ}}{d\ln P}\right) \quad (6.244)$$

  ici $\beta = P_{\rm gaz}/P$, $P_{\rm gaz}$ est la pression gazeuse, $P = P_{\rm gaz} + P_{\rm rad}$ est la pression totale, $P_{\rm rad} = a/3T^4$ est la pression de radiation et:

$$\mu^{-1} = \mu_i^{-1} + \mu_e^{-1} \sim 2X + \frac{3}{4}Y + \frac{Z}{2} = \frac{3+5X-Z}{4}, \;\; \nabla_\mu = \frac{4\pi R^4}{GM}\frac{P}{\mu}\frac{\partial\mu}{\partial X}\frac{\partial X}{\partial M}, \quad (6.245)$$

- soit le critère de Schwarzchild:

$$\nabla_{\rm ad}^* - \nabla_{\rm rad} < 0. \quad (6.246)$$

On a noté $\nabla_{\rm ad}^*$ le gradient corrigé de la pression turbulente:

$$\nabla_{\rm ad}^* = \frac{\Gamma_2 - 1}{\Gamma_2}\frac{{\rm d}\ln P_{\rm gaz}}{{\rm d}\ln P}. \quad (6.247)$$

Le calcul de $\frac{\varphi}{\delta}\nabla_\mu$ nécessite la connaissance des taux d'ionisation partielle des divers éléments. Ces quantités ne sont pas explicitement accessibles avec les équations d'état tabulées dont on dispose. Pour le critère de Ledoux on utilise l'approximation "gaz parfait avec radiation" de l'équation d'état (Cox & Giuli, 1968, Eq.3-30, p. 276) :

$$P = \frac{\rho\mathcal{R}T}{\mu} + \frac{a}{3}T^4 \quad (6.248)$$

mène à l'approximation fréquemment utilisée:

$$\frac{\varphi}{\delta} \sim \frac{\beta}{4-3\beta}. \quad (6.249)$$

Suivant la valeur `.TRUE.` ou `.FALSE.`, de la variable logique `ledoux` de la NAMELIST `NL_CONV` du fichier de données, l'un ou l'autre de ces deux critères est utilisé. Le critère de convection est formulé dans les routines `thermo`, *cf.* §7.80 (Page 229) `thermo_atm`, *cf.* §7.81 (Page 230) et `dgrad`, *cf.* §7.16 (Page 171). Dans la restitution de la partie convective de l'atmosphère, le gradient radiatif sera modifié afin d'assurer la continuité du gradient de température, *cf.* §6.3.3 (Page 69).

6.9.2 Calcul du gradient convectif

La routine conv_jmj, *cf.* §7.12.2 (Page 169), utilise un formalisme proche de celui de Henyey et al. (1965):

$$\nabla - \nabla^*_{\rm ad} = \frac{\Gamma(\Gamma+1)}{B},\ \nabla^*_{\rm ad} = \frac{\Gamma_2 - 1}{\Gamma_2}\frac{{\rm d}\ln P_{\rm gaz}}{{\rm d}\ln P},\ B = \xi\frac{l^4 g\delta(\rho c_{\rm p})^2}{H_{\rm p}K^2}, \tag{6.250}$$
$$K = \frac{4acT^3}{3\kappa\rho},\ g = \frac{GM}{R^2},\ \xi = \frac{1}{72}\left(3\frac{V}{Al}\right)^2\left(1+\frac{2Al}{3V\tau^2}\right)^2,$$

$P_{\rm gas}$ est la pression gazeuse i.e. thermodynamique + radiative et P est la pression totale i.e. gazeuse + turbulente, *cf.* §6.9.3 (Page 119), $P = P_{\rm gas} + P_{\rm tur}$, τ est l'épaisseur optique Rosseland de l'élément convectif:

$$\tau = \kappa\rho l = \kappa\rho\alpha H_{\rm p}, \tag{6.251}$$

$l = \alpha H_{\rm p}$ est la longueur de mélange, $H_{\rm p} = -\frac{{\rm d}R}{{\rm d}\ln P} = P/\rho g$ est l'échelle de hauteur de pression. Γ est l'efficacité de la convection, zéro réel de la cubique:

$$\Phi\Gamma^3 + \Gamma(\Gamma+1) = B(\nabla_{\rm rad} - \nabla^*_{\rm ad}), \tag{6.252}$$
$$\Phi = \varphi\left(1+\frac{2Al}{3V\tau^2}\right)^{-1},\ \varphi = \frac{3}{2}\left(3\frac{V}{Al}\right)^{-1}, \nabla_{\rm rad} = \frac{3}{16\pi acG}\frac{\kappa LP}{mT^4}.$$

La quantité V/A est le rapport du volume/surface de l'élément convectif. Le terme correctif $\frac{\partial\ln P_{\rm gaz}}{\partial\ln P}$ qui affecte le gradient adiabatique $\nabla_{\rm ad} \equiv \frac{\Gamma_2-1}{\Gamma_2}$ dans l'expression de ∇^*, *cf.* (6.250), est introduit pour avoir une estimation du gradient de température lors d'une transformation adiabatique avec pression turbulente, ce qui est très approximatif et physiquement mal établi.

En un point d'une zone convective, il semble irréaliste que la longueur de mélange puisse être supérieure à la distance qui sépare ce point de la plus proche limite de la zone convective. Ainsi que ? l'a proposé, dans la routine conv_a0, *cf.* §7.12.2 (Page 169), la longueur de mélange est prise égale à $l = \alpha_0 H_{\rm p}$ avec $\alpha_0 \equiv \alpha(1 - \nabla^*_{\rm ad}/\nabla_{\rm rad})$, la longueur de mélange utilisée devenant nulle à chaque limite RZ/CZ. Cette approximation est justifiée par le fait que les grandes incertitudes inhérentes à la théorie ne justifient pas d'introduire une trop grande rigueur dans les détails, *cf.* Cox & Giuli (1968). Par ailleurs, l'introduction de la distance exacte change la nature du problème, il devient intégro-différentiel ce qui complique considérablement la résolution numérique. Introduire la distance exacte à la plus proche limite n'est pas immédiat, du moins algorithmiquement, car le problème devient globalement implicite. Ce qu'évite la formulation locale d'Eggleton.

Dans conv_jmj et conv_a0 ces équations sont résolues avec $\xi = 1/162$, $\Phi = 9/4$ et $V/Al = 2/9$. La recherche du zéro de la cubique est faite à l'aide de l'algorithme de déflation.

La mise en œuvre de conv_jmj a bénéficié de la collaboration de J.Provost et de MJ.Goupil.

Dans la routine conv_cm, *cf.* §7.12.2 (Page 169), le gradient de température dans les zones convectives est calculé selon les prescriptions de Canuto & Mazzitelli (1991); l'expression (63) de ces auteurs, mise sous la forme implicite équivalente:

$$0 = \nabla + \Phi(\nabla - \nabla^*_{\rm ad}) - \nabla_{\rm rad} \tag{6.253}$$

est résolue itérativement. Φ est donné par Canuto & Mazzitelli (1991, eq. 32):

$$\Phi = a_1\Sigma^m[(1+a_2\Sigma)^n - 1]^p,\ \Sigma = 4A^2(\nabla - \nabla^*_{\rm ad}),\ A = \frac{l^2}{9\chi}\sqrt{\frac{g}{2H_{\rm p}}},\ \chi = \frac{K}{\rho c_{\rm p}} \tag{6.254}$$

ainsi que (5) et (6) avec $a_1 = 24.868$, $a_2 = 9.7666\times10^{-2}$, $m = 0.14972$, $n = 0.18931$ et $p = 1.8503$. Comme définie précédemment, la longueur de mélange est prise égale à $l = \alpha H_{\rm p}$. L'efficacité de la

convection est alors l'expression (8):

$$\Gamma = \frac{\sqrt{\Sigma + 1} - 1}{2}. \tag{6.255}$$

Dans la routine conv_cm_reza, *cf.* §7.12.2 (Page 169), similaire à conv_cm, il est tenu compte de la quantité thermodynamique $\delta = (\frac{\partial \ln \rho}{\partial \ln T})_{\rm P}$. Dans la routine conv_cgm_reza, *cf.* §7.12.2 (Page 169) le calcul du gradient convectif est effectué selon les prescriptions de Canuto et al. (1996) avec la prescription de Bernkopf (**?**) Ces deux routines ont été mises en œuvre et mises à la disposition des utilisateurs de Cesam2k20 par Reza Samadi, lesia, Observatoire de Paris

6.9.3 Pression turbulente

La théorie de la pression turbulente est très approximative; son expression est donnée par la relation phénoménologique:

$$P_{\rm turb} = \rho < v^2 >= a\rho v^2,\ v^2 = \frac{\Gamma}{1+\Gamma}\frac{\delta\alpha^2 P}{8\rho}(\nabla - \nabla^*_{\rm ad}), \tag{6.256}$$

où v est la "vitesse" des éléments convectifs; la valeur du coefficient phénoménologique a est mal déterminée, et varie suivant les auteurs, $a \sim 0.21$ pour Canuto & Mazzitelli (1991), $a \sim 0.50$ chez Henyey et al. (1965) ou encore, $a = \pi/2$.

Dans les conditions de la zone convective solaire, ce n'est que lorsque la température est inférieure à 50 000K que le rapport $P_{\rm turb}/P$ dépasse 1/10 000; ce rapport présente un maximum, de l'ordre de 16%, vers 8 000K, i.e. à l'intérieur de l'atmosphère. Cette correction de pression turbulente n'est supérieure à 1% que dans une zone localisée sous la surface où les températures se situent dans l'intervalle 6 500K, 15 500K; ce qui correspond au début de la zone d'ionisation de l'hydrogène.

En tenant compte de la pression turbulente, le problème devient totalement implicite; en effet, lorsque la pression totale, la température et la composition chimique sont connues, le calcul des grandeurs thermodynamiques nécessite la connaissance de la vitesse de convection, déduite elle même des grandeurs thermodynamiques calculées à partir de la pression gazeuse, donc de la pression totale et de la pression turbulente. Dans Cesam2k20 ce problème implicite est résolu en considérant la pression totale et la pression gazeuse comme deux variables dépendantes distinctes.

La mise en place de la pression turbulente a bénéficié de la collaboration de S. Brun.

Problem: *La correction empirique* $\frac{{\rm d}\ln P_{\rm gaz}}{{\rm d}\ln P}$ *affectant le gradient adiabatique rend le problème extrèmement instable. Dans Cesam2k20 cette correction est ignorée si la donnée* CPTURB *de la NAMELIST* NL_CONV *est positive ou nulle.*

Problem: *Cesam2k20 permet d'abandonner (resp. de réintroduire) la pression turbulente en cours et à fortiori au début d'une évolution, le nombre d'équations de structure effectivement résolues est alors 7, (resp. 8).*

6.9.4 Localisation des limites des zones convectives

Hors atmosphère, cette localisation a pour but essentiel de fixer les limites de la zone à mélanger i.e. par mélange convectif, afin de localiser avec précision les discontinuités qui en résultent. Dans l'atmosphère, cette localisation est sans objet puisque la composition chimique et la vitesse angulaires y sont supposées constantes.

Les limites des zones convectives sont déterminées dans la routine lim_zc, *cf.* §7.42 (Page 204). Dans un premier temps on localise en indice de couche, à l'aide du sous-programme dgrad, *cf.* §7.16 (Page 171), les intervalles où se produisent un changement de signe de la quantité $\nabla_{\rm rad} - \nabla^*_{\rm ad}$. Dans une seconde étape, la localisation de chaque changement de signe est affinée, par dichotomie, jusqu'à la précision définie par la variable loc_zc. Celle-ci ayant été initialisée dans la routine cesam, *cf.*

§7.7.3 (Page 158), suivant le type de précision requis. Ainsi, avec la précision `sa` *cf.* Table 3.1 (Page 34), la limite de la zone convective solaire sera déterminée à mieux que `loc_zc`=10^{-5} de la largeur de l'intervalle la contenant e.g. à mieux que $\sim 30\,\mathrm{m}$. C'est cette localisation améliorée qui sert de référence pour d'éventuelles extensions.

6.9.5 Extension des zones convectives

L'extension des zones convectives (`jpz=.FALSE.` dans la NAMELIST `NL_CONV`) et la pénétration convective (`jpz=.TRUE.`) ont pour effet d'étendre la zone de mélange et de fixer le gradient de température égal au gradient adiabatique dans les zones d'extension. Dans Cesam2k20 on distingue undershooting[18] i.e. extension de la limite vers le centre, et overshooting i.e. extension de la limite vers la surface; leur importance est fixée, respectivement, par les paramètres `ovshts` et `ovshti` de la NAMELIST `NL_CONV` du fichier de données; les zones convectives sont respectivement étendues de `ovshts`$\times H_\mathrm{p}$ et de `ovshti`$\times H_\mathrm{p}$. En ce qui concerne l'overshoot d'un cœur convectif de rayon R_c, l'extension de la zone mélangée est limitée à $\min($`ovshts`$\times H_\mathrm{p}, R_\mathrm{c}\times$ `ovshts`$)$ si `jpz=.FALSE.`; elle a pour valeur R_c, si `jpz=.TRUE.` Pour la pénétration convective, obtenue avec `jpz=.TRUE.`, l'extension de la zone mélangée L_p et du gradient adiabatiaque est déterminée par **?**:

$$L_\mathrm{p} = \frac{\zeta H_\mathrm{p}}{K_\mathrm{p}},\ K_\mathrm{p} = \frac{\partial \ln K}{\partial \ln P}, \tag{6.257}$$

$\zeta \sim 0.5$=`ovshti`, étant un paramètre ajustable.

Dans une extension le gradient est pris égal au gradient adiabatique lorsque le paramètre d'overshooting est positif, au gradient radiatif avec une valeur négative, *cf.* §3.8 (Page 22). La composition chimique et la vitesse angulaire sont homogénéisées, *cf.* §6.9.6 (Page 120)

Les dispositions suivantes sont prévues:

- La pénétration convective n'a d'action que sur les limites inférieures des zones convectives.
- La pénétration convective et les overshoots n'ont d'effet que si la température est supérieure[19] à 500 000K; cette disposition a pour but d'éviter "d'overshooter" les zones convectives superficielles.
- Avec la présente version de Cesam2k20, on ne peut donc pas étendre une limite RZ/CZ située dans l'atmosphère.

6.9.6 Mélange convectif

Dans les zones mélangées *i.e.* les zones convectives et leurs extensions par pénétration convective, la composition chimique et la vitesse anguliare sont homogénéisées. Avec diffusion on utilise un coefficient de diffusion de mélange turbulent, typiquement, $d_M = 10^{13}$ cm s^{-1}, ordre de grandeur obtenu comme produit de valeurs typiques de la vitesse de convection et de l'échelle de hauteur de pression. En l'absence de diffusion, le mélange est effectué par moyennes spatiales couplées avec les réactions thermonucléaires i.e. par la résolution numérique simultanée du mélange et de l'évolution nucléaire, ce qui constitue un problème intégro-différentiel, *cf.* §7.7.1 (Page 222).

6.9.7 Retrait d'un coeur convectif

L'étendue d'un coeur convectif varie au cours de l'évolution. Lors d'une extension la composition chimique, donc la densité, sont discontinues sur la limite externe du coeur. Lors d'une contraction,

[18] Au sens d'Oxford, undershooting et overshooting sont utilisés de façon incorrecte, ce qui provoque des réactions épidermiques, parfois violentes, de collègues de pure expression anglo-saxonne; en franglais ils ont le sens imagé d'extension inférieure et supérieure.

[19] Pour modifier cette valeur il faut intervenir dans la routine `lim_zc`.

la composition chimique devient continue de la zone de retrait jusqu'au centre; la dérivée spatiale du poids moléculaire moyen présente une discontinuité à la limite avec la zone radiative adjacente. Une description précise de ce retrait tenant compte des variations de composition chimique dues aux réactions thermonucléaires locales dans les parties devenant radiatives, est très complexe à traiter numériquement à cause des discrétisations spatiale et temporelle inévitables. Cesam2k20 suppose que les abondances des éléments chimiques varient linéairement dans la partie radiative correspondant au retrait du coeur. La situation similaire qui se produit lors de l'augmentation de l'abscisse de la base d'une zone convective n'est pas traitée dans la présente version de Cesam2k20. Les discontinuités fossiles[20] ne reçoivent pas de traitement particulier; étant ignorées la diffusion numérique est chargée de les faire lentement disparaître. Avec diffusion, le retrait des coeurs convectifs ne reçoivent pas de traitement spécifique avec la présente version de Cesam2k20.

6.9.8 Semi-convection

Dans les zones où le gradient radiatif vérifie:

$$\nabla_{\rm ad} < \nabla_{\rm rad} < \frac{\varphi}{\delta}\nabla_\mu \tag{6.258}$$

le milieu est *vibrationellement* instable, (**?**, §30.4.2, p.284). Le mélange des éléments chimique n'y est pas aussi efficace qu'en cas de l'instabilité convective. Cesam2k20 traite cette situation par diffusion, le coefficient de diffusion étant calculé selon les prescriptions de **?**.Dans les parties concernées le gradient est pris égal au gradient radiatif.

Problem: *Sans diffusion, la semi-convection n'est pas prise en compte.*

6.9.9 Estimation de la fréquence de v

Dans les fichiers de sortie en ASCII, Cesam2k20 donne la valeur de la quantité:

$$A = \frac{1}{\Gamma_1}\frac{\partial \ln P}{\partial \ln R} - \frac{\partial \ln \rho}{\partial \ln R} \tag{6.259}$$

pour chaque couche du modèle. A permet l'estimation de la fréquence de $\mathbf{v}$; A, différence de deux dérivées premières, est équivalente à une dérivée seconde. Le calcul de la fréquence de $\mathbf{v}$ est donc sensible aux discontinuités de la dérivée première de la composition chimique qui provoquent des oscillations; on a indiqué au §5.3 (Page 43), que la prise en compte d'une légère diffusion turbulente permettait d'obtenir des profils quasi théoriques pour la fréquence de $\mathbf{v}$.

Sur une limite RZ/CZ, la composition chimique est une fonction discontinue si on ignore la diffusion microscopique. Avec diffusion, la densité est continue, non dérivable, A se comporte alors comme une fonction δ. Pour décrire finement le comportement de la fréquence de $\mathbf{v}$, dans les fichiers de sortie destinés aux exploitations sismologiques, *cf.* §7.58 (Page 214), des points sont ajoutés au voisinage de ces discontinuités, *cf.* §7.19 (Page 172). La dérivée spatiale de la densité étant accessible dans Cesam2k20, l'équation 6.259 (Page 121) est utilisée pour le calcul de la fréquence de $\mathbf{v}$. Pour que cette relation soit aussi appliquable il est nécessaire que l'approximation de la densité soit numériquement satisfaisante; ce qui n'est réalisé que lorsque la densité est continue aux limites zones radiatives / zones convectives. ***Cette formulation n'est donc appliquable que si la diffusion des éléments chimiques est prise en compte.*** Si tel n'est pas le cas, comme par exemple avec la précision `np` *cf.* Table 3.1 (Page 34) ou sans diffusion.

Dans ces cas une formulation approchée est utilisée; elle repose sur une approximation de $\varphi =$

[20] Discontinuités de composition chimiques des pas temporels précédents.

$\frac{\partial \ln \rho}{\partial \ln \mu}$:

$$d\ln\rho = \alpha\, d\ln P - \delta\, d\ln T + \varphi\, d\ln\mu \Longrightarrow$$
$$\frac{\partial \ln \rho}{\partial \ln P} = \alpha - \delta\frac{\partial \ln T}{\partial \ln P} + \varphi\frac{\partial \ln \mu}{\partial \ln P} = \alpha - \delta\nabla + \varphi\nabla_\mu$$

avec $\nabla_\mu = \frac{\partial \ln \mu}{\partial \ln P}$. φ n'est donné ni par les routines ni par les tabulations d'Equation of State. L'approximation "gaz parfait avec radiation" de l'équation d'état (Cox & Giuli, 1968, Eq.3-30, p. 276) :

$$P = \frac{\rho \mathcal{R} T}{\mu} + \frac{a}{3}T^4 \tag{6.260}$$

mène à l'approximation fréquemment utilisée:

$$\frac{\varphi}{\delta} \sim \frac{\beta}{4-3\beta}. \tag{6.261}$$

En tenant compte de:

$$\frac{1}{\Gamma_1} = \alpha - \delta\nabla_{\rm ad},\ H_p = -\frac{\partial R}{\partial \ln P} \tag{6.262}$$

on obtient la relation utilisée par Cesam2k20:

$$A = \frac{R\delta}{H_p}\left(\nabla - \nabla_{\rm ad} + \frac{\beta}{4-3\beta}\nabla_\mu\right) \tag{6.263}$$

Toutefois la formulation utilisée dans les versions précédentes de CESAM reste disponible. Pour la restituer, dans la routine `cesam.f90`, *cf.* §7.7.3 (Page 158), du sous-directory `SOURCE`, ligne 225, affecter `.FALSE.` au paramètre `new_bv`. L'expression suivante sera utilisée:

$$A = \frac{R\delta}{H_{\rm p}}\left(\nabla_{\rm ad} - \nabla\right) - 4\pi R^3 \sum_i \frac{\partial \rho}{\partial X_i}\frac{\partial X_i}{\partial M}. \tag{6.264}$$

Cette formulation présente des inconvénients:

- Elle nécessite les dérivées de la densité par rapport aux différentes espèces chimiques, quantités inaccessibles avec les équations d'état tabulées, si bien que la somme sur les espèces chimiques est nécessairement réduite au seul élément H (i.e. $i \equiv 1$). Une conséquence est que (6.259) conduit à des valeurs systématiquement inférieures à celles données par (6.264).
- Un autre inconvénient est la non dérivabilité de la densité aux limites zone radiative/zone convective. Comme d'une part, ces limites ne peuvent coïncider *parfaitement* avec les points de grille et que, d'autre part, les équations d'évolution de la composition chimique ne sont pas résolues simultanément avec les équations de la structure, il se produit nécessairement un petit décalage entre la discontinuité de la composition chimique, donc de la densité (ou seulement de leurs gradients), et le point limite où l'égalité des gradients est réalisée. De ces inconsistances, il résulte des oscillations, parfois violentes des valeurs de A obtenues par (6.259) et, de là, de la fréquence de $\mathbf{v}$.

Toutefois le gradient de composition chimique étant supposé nul dans l'atmosphère restituée, ces difficultés ne s'y présentent pas et (6.264) est utilisée.

On présente une autre formulation, qui bien qu'implantée dans Cesam2k20, n'est pas utilisée. Comme (6.259), elle nécessite une oscularité suffisante des variables. La pression P, le rayon, la

masse, étant approchés par une spline en fonction de la variable d'indice q, on a accès à leurs dérivées premières et secondes. L'idée est d'en déduire le gradient de densité puisque:

$$\rho = -\frac{1}{g}\frac{\partial P}{\partial R},\ g = \frac{GM}{R^2}. \tag{6.265}$$

Ainsi on tiendra compte *implicitement* et de façon cohérente, à travers la formulation numérique,des détails de l'équation d'état et des discontinuités. On a:

$$\frac{\partial \ln \rho}{\partial \ln R} = \frac{R}{\rho}\frac{\partial \rho}{\partial R} = \frac{R}{\rho g}\left(\frac{1}{g}\frac{\partial g}{\partial R}\frac{\partial P}{\partial R} - \frac{\partial^2 P}{\partial R^2}\right), \tag{6.266}$$

$$\frac{\partial g}{\partial R} = \frac{G}{R^2}\frac{\partial M}{\partial R} - \frac{2g}{R} = 4\pi\rho G - \frac{2g}{R}. \tag{6.267}$$

A des fins de référence et sous une forme adaptée à la programmation, on donne ci-après les expressions des deux termes de A ((6.259)) en fonction des quantités ξ, ζ et μ et de leurs dérivées par rapport à la variable d'espacement q.

- Avec les variables $\xi = \ln P$, $\zeta = \left(\frac{R}{R_\odot}\right)^2$ et$\mu = \left(\frac{M}{M_\odot}\right)^{2/3}$:

$$\frac{\partial \zeta}{\partial R} = \frac{2R}{R_\odot^2} = \frac{2\sqrt{\zeta}}{R_\odot},\ \frac{\partial}{\partial R} = \frac{\partial q}{\partial \zeta}\frac{\partial \zeta}{\partial R}\frac{\partial}{\partial q} = \frac{2\sqrt{\zeta}}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}, \tag{6.268}$$

$$\frac{\partial \ln P}{\partial R} = \frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}\frac{\partial \zeta}{\partial R} = \frac{2\sqrt{\zeta}}{R_\odot}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}, \tag{6.269}$$

$$\frac{\partial \ln P}{\partial \ln R} = R\frac{\partial \ln P}{\partial R} = 2\zeta\left[\frac{\partial \xi}{\partial q}\Big/\frac{\partial \zeta}{\partial q}\right]. \tag{6.270}$$

$$\frac{\partial P}{\partial R} = P\frac{\partial \ln P}{\partial R} = \frac{2RP}{R_\odot^2}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}, \tag{6.271}$$

$$\begin{aligned}\frac{\partial}{\partial R}\frac{\partial q}{\partial \zeta} &= \frac{\partial}{\partial R}\left(\frac{\partial \zeta}{\partial q}\right)^{-1} = -\left(\frac{\partial \zeta}{\partial q}\right)^{-2}\frac{\partial}{\partial R}\frac{\partial \zeta}{\partial q} \\ &= -\left(\frac{\partial \zeta}{\partial q}\right)^{-2}\frac{2\sqrt{\zeta}}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}\frac{\partial \zeta}{\partial q} = -\frac{2\sqrt{\zeta}}{R_\odot}\left(\frac{\partial \zeta}{\partial q}\right)^{-3}\frac{\partial^2 \zeta}{\partial q^2},\ nonumber\end{aligned} \tag{6.272}$$

$$\frac{\partial}{\partial R}\frac{\partial \ln P}{\partial q} = \frac{2\sqrt{\zeta}}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}\frac{\partial \ln P}{\partial q} = \frac{2\sqrt{\zeta}}{R_\odot}\left(\frac{\partial \zeta}{\partial q}\right)^{-1}\frac{\partial^2 \ln P}{\partial q^2}, \tag{6.273}$$

$$\begin{aligned}\frac{\partial^2 P}{\partial R^2} &= \frac{\partial}{\partial R}\frac{\partial P}{\partial R} = \frac{\partial}{\partial R}\left\{\frac{2RP}{R_\odot^2}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}\right\} \\ &= \frac{1}{R}\frac{\partial P}{\partial R} + \frac{1}{P}\left(\frac{\partial P}{\partial R}\right)^2 + \left[\frac{\partial P}{\partial R}\Big/\frac{\partial q}{\partial \zeta}\right]\frac{\partial}{\partial R}\frac{\partial q}{\partial \zeta} + \left[\frac{\partial P}{\partial R}\Big/\frac{\partial \ln P}{\partial q}\right]\frac{\partial}{\partial R}\frac{\partial \ln P}{\partial q}\ nonumber \\ &= \frac{\partial P}{\partial R}\left\{\frac{1}{R} + \frac{1}{P}\frac{\partial P}{\partial R} + \frac{2R}{R_\odot^2}\left[\frac{\partial^2 \xi}{\partial q^2}\Big/\frac{\partial \xi}{\partial q} - \frac{\partial^2 \zeta}{\partial q^2}\Big/\frac{\partial \zeta}{\partial q}\right]\Big/\frac{\partial \zeta}{\partial q}\right\}\end{aligned} \tag{6.274, 6.275}$$

L'expression de A donnée par (6.259) est obtenue par (6.270) et (6.266) calculée à l'aide de (6.267), (6.271) et (6.275).

- Avec les variables $\xi = \ln P$, $\zeta = \frac{R}{R_\odot}$ et $\mu = \frac{M}{M_\odot}$:

$$\frac{\partial \zeta}{\partial R} = \frac{1}{R_\odot}, \quad \frac{\partial}{\partial R} = \frac{\partial q}{\partial \zeta}\frac{\partial \zeta}{\partial R}\frac{\partial}{\partial q} = \frac{1}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}, \tag{6.276}$$

$$\frac{\partial \ln P}{\partial R} = \frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}\frac{\partial \zeta}{\partial R} = \frac{1}{R_\odot}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}, \tag{6.277}$$

$$\frac{\partial \ln P}{\partial \ln R} = R\frac{\partial \ln P}{\partial R} = \frac{R}{R_\odot}\left[\frac{\partial \xi}{\partial q}\Big/\frac{\partial \zeta}{\partial q}\right]. \tag{6.278}$$

$$\frac{\partial P}{\partial R} = P\frac{\partial \ln P}{\partial R} = \frac{P}{R_\odot}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}, \tag{6.279}$$

$$\begin{aligned}\frac{\partial}{\partial R}\frac{\partial q}{\partial \zeta} &= \frac{\partial}{\partial R}\left(\frac{\partial \zeta}{\partial q}\right)^{-1} = -\left(\frac{\partial \zeta}{\partial q}\right)^{-2}\frac{\partial}{\partial R}\frac{\partial \zeta}{\partial q} \\ &= -\left(\frac{\partial \zeta}{\partial q}\right)^{-2}\frac{1}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}\frac{\partial \zeta}{\partial q} = -\frac{1}{R_\odot}\left(\frac{\partial \zeta}{\partial q}\right)^{-3}\frac{\partial^2 \zeta}{\partial q^2},\end{aligned}$$

$$\frac{\partial}{\partial R}\frac{\partial \ln P}{\partial q} = \frac{1}{R_\odot}\frac{\partial q}{\partial \zeta}\frac{\partial}{\partial q}\frac{\partial \ln P}{\partial q} = \frac{1}{R_\odot}\left(\frac{\partial \zeta}{\partial q}\right)^{-1}\frac{\partial^2 \ln P}{\partial q^2}, \tag{6.280}$$

$$\begin{aligned}\frac{\partial^2 P}{\partial R^2} &= \frac{\partial}{\partial R}\frac{\partial P}{\partial R} = \frac{\partial}{\partial R}\left\{\frac{P}{R_\odot}\frac{\partial \ln P}{\partial q}\frac{\partial q}{\partial \zeta}\right\} \\ &= \frac{1}{P}\left(\frac{\partial P}{\partial R}\right)^2 + \left[\frac{\partial P}{\partial R}\Big/\frac{\partial q}{\partial \zeta}\right]\frac{\partial}{\partial R}\frac{\partial q}{\partial \zeta} + \left[\frac{\partial P}{\partial R}\Big/\frac{\partial \ln P}{\partial q}\right]\frac{\partial}{\partial R}\frac{\partial \ln P}{\partial q} \\ &= \frac{\partial P}{\partial R}\left\{\frac{1}{P}\frac{\partial P}{\partial R} + \frac{1}{R_\odot}\left[\frac{\partial^2 \xi}{\partial q^2}\Big/\frac{\partial \xi}{\partial q} - \frac{\partial^2 \zeta}{\partial q^2}\Big/\frac{\partial \zeta}{\partial q}\right]\Big/\frac{\partial \zeta}{\partial q}\right\}.\end{aligned} \tag{6.281}$$

L'expression de A donnée par (6.259), est obtenue par (6.278) et (6.266) calculée à l'aide de (6.267), (6.279) et (6.281).

Problem: *Ces dernières relations doivent être adaptées si on tient compte de la rotation.*

Problem: *Des tests héliosismologiques ont mis en évidence que ces relations, bien que formellement plus cohérentes, faisaient ressortir les incohérences entre équation d'état et opacités. Il convient de ne les utiliser qu'avec prudence.*

6.10 Les réactions thermonucléaires

6.10.1 Abondances initiales

Pour éviter des calculs inutiles, il est nécessaire d'optimiser le réseau nucléaire utilisé en fonction de la connaissance, à priori, de la physique des modèles à calculer. Il est par exemple inutile de tenir compte des réactions thermonucléaires concernant le silicium si la température maximum n'atteint pas le seuil du déclanchement de ces réactions. Il est donc inutile de suivre explicitement l'évolution des éléments chimiques qui ne sont pas concernés par le réseau nucléaire retenu, sauf, évidemment, si on

désire en suivre la diffusion. Les isotopes de la mixture initiale n'étant pas retenus comme éléments chimiques ils ne doivent pas pour autant être totalement ignorés, d'autant que leur contribution à la masse totale, quelques 10^{-3}, est loin d'être négligeable. Dans Cesam2k20, ces éléments sont représentés par l'isotope *fictif* dont la masse (resp. charge) égale la moyenne pondérée des masses (resp. charges) des éléments dont il n'est pas tenu compte explicitement. Par exemple cet isotope fictif, ici noté Ex, est souvent ^{24}Si avec une mixture initiale solaire. Cet isotope fictif complémentaire a l'indice `i_ex`, *cf.* §7.2 (Page 135).

L'abondance initiale en *nombre par mole* de chaque isotope de la mixture utilisée dans Cesam2k20 est déterminée en fonction du réseau de réactions retenu, *cf.* §?? (Page ??), à partir:

- Des abondances initiales *en masse* de l'hydrogène X, et de l'hélium Y, lues dans la NAMELIST `NL_CHIM` du fichier de données, *cf.* §3.3 (Page 14).
- Des proportions initiales *en nombre* des éléments lourds dans $Z = 1 - X - Y$.
- Des rapports isotopiques initiaux.

De l'ensemble de ces données résulte un système d'équations linéaires dont la solution est la composition chimique initiale. Dans l'exemple qui suit, x_i désigne l'abondance par mole i.e. l'abondance par masse divisée par la masse atomique ν_i, de l'isotope d'ordre $i = 1, \ldots, 10$ dans la liste: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, Ø16, Ø17 et Ex. Les rapports d'abondances initiaux en nombre, dans Z, du carbone, azote et oxygène seront respectivement désignés par C, N et O; par exemple pour le soleil: $C = 0.24551,\ N = 0.06458,\ O = 0.51295$. Les rapports isotopiques initiaux en nombre seront désignés par le rapport de leurs symboles e.g. ^{3}He/^{4}He$= 4.2\,10^{-4}$. Les abondances en nombre x_4 et x_5 des isotopes ^{12}C et ^{13}C du carbone vérifient:

$$C = \frac{n_C}{n_Z} = \frac{x_4 + x_5}{\Sigma_{i=4}^{10} x_i}, \frac{x_5}{x_4} = {}^{13}\mathrm{C}/{}^{12}\mathrm{C} \tag{6.282}$$

On a respectivement noté n_C et n_Z les nombres totaux d'isotopes du carbone et d'éléments lourds. Des équations similaires valent pour tous les éléments du réseau nucléaire. Le système de 10 équations linéaires vérifiées par les 10 valeurs des abondances initiales en nombre, $x_{i,\ i=1,\ldots,10}$ s'écrit:

$$\begin{aligned}
x_1\nu_1 &= X \\
x_2\nu_2 + x_3\nu_3 &= Y \\
\Sigma_{i=4}^{10} x_i\nu_i &= 1 - X - Y \\
x_4 + x_5 - C \times \Sigma_{i=4}^{10} x_i &= 0 \\
x_6 + x_7 - N \times \Sigma_{i=4}^{10} x_i &= 0 \\
x_8 + x_9 - O \times \Sigma_{i=4}^{10} x_i &= 0 \\
x_2 - x_3\ {}^3\mathrm{He}/{}^4\mathrm{He} &= 0 \\
x_5 - x_4\ {}^{13}\mathrm{C}/{}^{12}\mathrm{C} &= 0 \\
x_7 - x_6\ {}^{15}\mathrm{N}/{}^{14}\mathrm{N} &= 0 \\
x_9 - x_8\ {}^{17}\mathrm{O}/{}^{16}\mathrm{O} &= 0
\end{aligned} \tag{6.283}$$

Pour chaque isotope, on fixe une abondance minimale au dessous de laquelle l'isotope est considéré comme inexistant, son évolution temporelle n'est alors pas contrôlée. La plupart des rapports isotopiques utilisés sont ceux de la nébuleuse solaire issus de la table 3 des nuclides de l'article de ?. Le rapport isotopique ^{3}He/^{4}He$= 1.104 \times 10^{-4}$ est déduit d'une mesure dans Jupiter; toutefois si ^{2}H est pris à l'équilibre, il convient d'ajouter à l'abondance initiale de ^{3}He celle de ^{2}H qui est transformée en ^{3}He pendant la PMS, d'où la valeur du rapport isotopique initial ^{3}He/^{4}He$= 4.2\,10^{-4}$.

L'ordre des éléments chimiques est en général celui des masses croissantes, l'isotope fictif venant en dernier, mais ce n'est pas une règle intangible. L'ordre est fixé dans la routine `tabul_nuc`, *cf.* §7.77

(Page 227), e.g. dans ppcno3a9 l'ordre des éléments est: ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N, ^{16}O, ^{17}O, Ex. L'isotope fictif venant en dernier.

6.10.2 Cycle PP simplifié

Il s'agit de la routine nuc_gong, *cf.* § ?? (Page ??), du Solar Model Comparison Project de Christensen-Dalsgaard (1988). On ne tient compte que d'un seul élément chimique: l'hydrogène dont l'abondance par unité de masse est ici notée X. L'énergie thermonucléaire générée par unité de masse et de temps est:

$$\epsilon = Q_e a_{11} \frac{X^2}{2m_u} \rho T_9^{-\frac{2}{3}} \exp(-bT_9^{-\frac{1}{3}}) \tag{6.284}$$

qui correspond à un taux:

$$\dot{X} = \frac{\partial X}{\partial t} = -R_e a_{11} \frac{X^2}{2} \rho T_9^{-\frac{2}{3}} \exp(-bT_9^{-\frac{1}{3}}) \tag{6.285}$$

où $R_e = 6.5$, $Q_e = 6.5 \times 10^{-5}$, $a_{11} = 4.21 \times 10^{-15}$, $b = 3.6$, m_u est l'unité de masse atomique et $T_9 = T \times 10^{-9}$.

6.10.3 Exemple de réseau nucléaire: cycles PP, CNO et 3α

A titre d'exemple, on détaille la construction d'un réseau nucléaire rassemblant les cycles PP, CNO et 3α

Pour simplifier les notations, on désigne indistinctement par un même symbole l'isotope et son abondance par mole. Les principales réactions thermonucléaires des cycles PP, CNO et 3α sont les suivantes (Clayton, 1968, p. 380,390):

$$\begin{array}{ll}
\underbrace{\begin{array}{ll}
R_1 & : \ ^{1}\mathrm{H}(p,\beta^+\nu)^{2}\mathrm{H} \\
R_2 & : \ ^{2}\mathrm{H}(p,\gamma)^{3}\mathrm{He} \\
R_3 & : \ ^{3}\mathrm{He}(^{3}\mathrm{He},2p)^{4}\mathrm{He} \\
R_4 & : \ ^{4}\mathrm{He}(^{3}\mathrm{He},\gamma)^{7}\mathrm{Be} \\
R_5 & : \ ^{7}\mathrm{Li}(p,\alpha)^{4}\mathrm{He} \\
R_6 & : \ ^{7}\mathrm{Be}(e^-\nu\gamma)^{7}\mathrm{Li} \\
R_7 & : \ ^{7}\mathrm{Be}(p,\gamma)^{8}\mathrm{B}^*(\beta^+\nu)^{8}\mathrm{Be}(\alpha)^{4}\mathrm{He}
\end{array}}_{\text{cycle PP}} &
\underbrace{\begin{array}{ll}
R_8 & : \ ^{12}\mathrm{C}(p,\gamma)^{13}\mathrm{N}(\beta^+\nu)^{13}\mathrm{C} \\
R_9 & : \ ^{13}\mathrm{C}(p,\gamma)^{14}\mathrm{N} \\
R_{10} & : \ ^{14}\mathrm{N}(p,\gamma)^{15}\mathrm{O}(\beta^+\nu)^{15}\mathrm{N} \\
R_{11} & : \ ^{15}\mathrm{N}(p,\gamma)^{16}\mathrm{O} \\
R_{12} & : \ ^{15}\mathrm{N}(p,\alpha)^{12}\mathrm{C} \\
R_{13} & : \ ^{16}\mathrm{O}(p,\gamma)^{17}\mathrm{F}(\beta^+\nu)^{17}\mathrm{O} \\
R_{14} & : \ ^{17}\mathrm{O}(p,\alpha)^{14}\mathrm{N}
\end{array}}_{\text{cycle CNO}}
\end{array} \tag{6.286}$$

$$\underbrace{\begin{array}{ll}
R_{15} & : \ ^{4}\mathrm{He}(2\alpha,\gamma)^{12}\mathrm{C} \\
R_{16} & : \ ^{12}\mathrm{C}(\alpha,\gamma)^{16}\mathrm{O} \\
R_{17} & : \ ^{16}\mathrm{O}(\alpha,\gamma)^{20}\mathrm{Ne}
\end{array}}_{\text{cycle } 3\alpha} \tag{6.287}$$

Compte non tenu de l'effet d'écran, le nombre R_i de réactions i, $i = 1, \ldots, 17$, par sec^{-1} g^{-1} est obtenu en mutipliant le taux tabulé par:

- ρ pour les réactions binaires entre éléments différents,
- $\rho/2!$ pour les réactions binaires entre éléments identiques: R_1 et R_3,
- $\rho^2/3!$ pour la réaction ternaire R_{15}, ?, p. 86-87.

Le nombre d'électrons par mole e^- et les taux de réaction incluant l'effet d'écran sont des fonctions $R_{i,\,i=1,\ldots,17}(T,\rho,\mathcal{X})$, $\mathcal{X}$ désignant le vecteur de composition chimique. Ils sont calculés ainsi que leur dérivées, par rapport à $(T,\rho,\mathcal{X})$, dans la routine `rq_reac`, en utilisant les tables construites par le programme `tab_reac`.

Bien que les réactions nucléaires ne soient pas uniquement localisées dans un milieu totalement ionisé, e.g. le fer n'est pas totalement ionisé au centre du soleil, le nombre d'électrons libres par mole est pris égal à:

$$n_{\rm e} = \sum_i z_i x_i. \tag{6.288}$$

6.10.4 Eléments à l'équilibre

Dans certaines phases de l'évolution, des éléments e.g. ^{2}H, ^{7}Li et ^{7}Be pour le soleil, ont des temps caractéristiques d'évolution très courts par rapport à l'échelle de temps de l'évolution. Leur taux de création est égal à leur taux de destruction. De tels éléments sont dits "***à l'équilibre***". L'abondance de chaque élément à l'équilibre résulte directement de la relation d'équilibre, ce qui permet des simplifications importantes. ^{2}H, ^{7}Li et ^{7}Be sont considérés à l'équilibre dans la routine `ppcno3a9`. On a alors:

$$0 = \frac{{\rm d}\ {}^2{\rm H}}{{\rm d}t} = R_1\ {}^1{\rm H}^2 - R_2\ {}^1{\rm H}\ {}^2{\rm H} \Rightarrow\ {}^2{\rm H} = \frac{R_1\ {}^1{\rm H}}{R_2}, \tag{6.289}$$

$$\left\{\begin{array}{l} 0 = \frac{{\rm d}\ {}^7{\rm Be}}{{\rm d}t} = R_4\ {}^3{\rm He}\ {}^4{\rm He} - R_6\ {}^7{\rm Be}\ e^- - R_7\ {}^1{\rm H}\ {}^7{\rm Be} \\ 0 = \frac{{\rm d}\ {}^7{\rm Li}}{{\rm d}t} = R_6\ {}^7{\rm Be}\ e^- - R_5\ {}^1{\rm H}\ {}^7{\rm Li} \end{array}\right. \Rightarrow \left\{\begin{array}{l} {}^7{\rm Be} = \frac{R_4\ {}^3{\rm He}\ {}^4{\rm He}}{R_6 e^- + R_7\ {}^1{\rm H}} \\ {}^7{\rm Li} = \frac{R_6\ {}^7{\rm Be}\ e^-}{R_5\ {}^1{\rm H}} \\ R_4\ {}^3{\rm He}\ {}^4{\rm He} = R_5\ {}^1{\rm H}\ {}^7{\rm Li} + R_7\ {}^1{\rm H}\ {}^7{\rm Be} \end{array}\right. \tag{6.290}$$

6.10.5 Effet d'écran

L'effet d'écran est un abaissement de la barrière de potentiel coulombienne dû à la charge d'espace des ions. Il se traduit par l'augmentation des taux des réactions thermonucléaires. Pour une évolution jusque sur la branche des géantes, l'approximation "*écran faible*" est suffisante, son expression est la suivante (Clayton, 1968, eq. 4.215-221):

$$f = \exp(-\frac{U_0}{kT}) = \exp\left(0.188 z_1 z_2 \sqrt{\frac{\rho\ \zeta}{T_6^3}}\right), \qquad \zeta = \sum_i z_i(1+z_i)x_i \tag{6.291}$$

z_i étant la charge du noyau i, x_i l'abondance, par mole, de l'élément chimique i et $T_6 = T \times 10^{-6}$. ζ représente l'effet des charges. L'approximation "*écran faible*", n'est valable que si f est proche de l'unité.

Cesam2k20 offre l'alternative d'utiliser l'écrantage de Mitler (1977), qui généralise, d'après cet auteur, les cas d'écrantage faible et intermédiaire; le cas écrantage fort n'est pas implémenté dans la version actuelle de Cesam2k20. Pour son emploi, coder `mitler=.TRUE.` dans la NAMELIST `NL_NUC` du fichier de données, *cf.* §3.3 (Page 14). L'installation de l'écrantage de Mitler a bénéficié d'une collaboration avec S. Turck-Chièze.

Problem: *Pour la capture électronique de la réaction $^7Be(e^-\nu\gamma)^7Li$, l'effet d'écran est inclus dans l'expression du taux de réaction (?).*

6.10.6 Energie thermonucléaire et neutrinos

La quantité d'énergie thermonucléaire, $Q_i, i = 1, \ldots, 17$, libérée par chaque réaction i, est déterminée pour chaque réaction par le programme `tab_reac` à partir des excès de masse et des énergies des neutrinos, Clayton (1968, p. 289). Ces quantités sont tabulées avec les taux des réactions.

La quantité d'énergie thermonucléaire libérée, est alors:$\sum_i Q_i R_i$ e.g. pour les cycles PP, CNO et 3 α, on a:

$$\begin{array}{l}\epsilon = q_1\ {}^1\mathrm{H}^2 + q_2\ {}^1\mathrm{H}\ {}^2\mathrm{H} + q_3\ {}^3\mathrm{He}^2 + q_4\ {}^3\mathrm{He}\ {}^4\mathrm{He} + q_5\ {}^1\mathrm{H}\ {}^7\mathrm{Li} + (q_6\ e^- + q_7\ {}^1\mathrm{H})\ {}^7\mathrm{Be} \\ +q_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C} + q_9\ {}^1\mathrm{H}\ {}^{13}\mathrm{C} + q_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N} + (q_{11} + q_{12}){}^1\mathrm{H}\ {}^{15}\mathrm{N} + q_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O} + q_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O} \\ +q_{15}\ {}^4\mathrm{He}^3 + q_{16}\ {}^4\mathrm{He}\ {}^{12}\mathrm{C} + q_{17}\ {}^4\mathrm{He}\ {}^{16}\mathrm{O},\end{array} \tag{6.292}$$

chacune des lignes représente l'énergie libérée respectivement par les cycles PP, CNO et 3α. Ces quantités apparaissent séparément dans la liste du modèle. Avec ^{2}H, ^{7}Li et ^{7}Be à l'équilibre, il convient d'utiliser les valeurs à l'équilibre établies précédemment.

Le nombre de neutrinos, par unité de masse et de temps, émis respectivement par les réactions R_1, R_6, R_7, R_8, R_{10} et R_{13} est:

$$\begin{array}{l}H_n = R_1\ {}^1\mathrm{H}^2,\ Be_n = R_6 e^-\ {}^7\mathrm{Be},\ B_n = R_7\ {}^1\mathrm{H}\ {}^7\mathrm{Li}, \\ N_n = R_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C},\ O_n = R_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N},\ F_n = R_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O}.\end{array} \tag{6.293}$$

Le nombre de neutrinos reçus sur terre par les expériences du chlore et du galium est calculé en supposant la source distante d' une unité astronomique

6.10.7 Equations d'évolution

Pour l'exemple des cycles PP+CNO+3α, avec ^{2}H, ^{7}Be, et ^{7}Li à l'équilibre, en désignant les abondances par mole par les symboles isotopiques, les équations d'évolution sont les suivantes:

$$
\begin{aligned}
\frac{\mathrm{d}^1\mathrm{H}}{\mathrm{d}t} &= -3R_1\ {}^1\mathrm{H}^2 + 2R_3\ {}^3\mathrm{He}^2 - R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He} \\
&\quad -R_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C} - R_9\ {}^1\mathrm{H}\ {}^{13}\mathrm{C} - R_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N} - (R_{11}+R_{12})^1\mathrm{H}\ {}^{15}\mathrm{N} \\
&\quad -R_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O} - R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O}, \\
\frac{\mathrm{d}^3\mathrm{He}}{\mathrm{d}t} &= R_1\ {}^1\mathrm{H}^2 - 2R_3\ {}^3\mathrm{He}^2 - R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He}, \\
\frac{\mathrm{d}^4\mathrm{He}}{\mathrm{d}t} &= R_3\ {}^3\mathrm{He}^2 + R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He} \\
&\quad +R_{12}\ {}^1\mathrm{H}\ {}^{15}\mathrm{N} + R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O} \\
&\quad -3R_{15}\ {}^4\mathrm{He}^3 - R_{16}\ {}^4\mathrm{He}\ {}^{12}\mathrm{C} - R_{17}\ {}^4\mathrm{He}\ {}^{16}\mathrm{O}, \\
\frac{\mathrm{d}^{12}\mathrm{C}}{\mathrm{d}t} &= -R_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C} + R_{12}\ {}^1\mathrm{H}\ {}^{15}\mathrm{N} \\
&\quad +R_{15}\ {}^4\mathrm{He}^3 - R_{16}\ {}^4\mathrm{He}\ {}^{12}\mathrm{C}, \\
\frac{\mathrm{d}^{13}\mathrm{C}}{\mathrm{d}t} &= R_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C} - R_9\ {}^1\mathrm{H}\ {}^{13}\mathrm{C}, \\
\frac{\mathrm{d}^{14}\mathrm{N}}{\mathrm{d}t} &= R_9\ {}^1\mathrm{H}\ {}^{13}\mathrm{C} - R_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N} + R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O}, \\
\frac{\mathrm{d}^{15}\mathrm{N}}{\mathrm{d}t} &= R_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N} - (R_{11}+R_{12})^1\mathrm{H}\ {}^{15}\mathrm{N}, \\
\frac{\mathrm{d}^{16}\mathrm{O}}{\mathrm{d}t} &= R_{11}\ {}^1\mathrm{H}\ {}^{15}\mathrm{N} - R_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O} \\
&\quad +R_{16}\ {}^4\mathrm{He}\ {}^{12}\mathrm{C} - R_{17}\ {}^4\mathrm{He}\ {}^{16}\mathrm{O}, \\
\frac{\mathrm{d}^{17}\mathrm{O}}{\mathrm{d}t} &= R_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O} - R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O}, \\
\frac{\mathrm{d}\,\mathrm{Ex}}{\mathrm{d}t} &= -\frac{1}{m_{\mathrm{Ex}}}\left[m_{{}^1\mathrm{H}}\frac{\mathrm{d}^1\mathrm{H}}{\mathrm{d}t} + m_{{}^3\mathrm{He}}\frac{\mathrm{d}^3\mathrm{He}}{\mathrm{d}t} + m_{{}^4\mathrm{He}}\frac{\mathrm{d}^4\mathrm{He}}{\mathrm{d}t} + m_{{}^{12}\mathrm{C}}\frac{\mathrm{d}^{12}\mathrm{C}}{\mathrm{d}t} + m_{{}^{13}\mathrm{C}}\frac{\mathrm{d}^{13}\mathrm{C}}{\mathrm{d}t}\right. \\
&\quad \left. +m_{{}^{14}\mathrm{N}}\frac{\mathrm{d}^{14}\mathrm{N}}{\mathrm{d}t} + m_{{}^{15}\mathrm{N}}\frac{\mathrm{d}^{15}\mathrm{N}}{\mathrm{d}t} + m_{{}^{16}\mathrm{O}}\frac{\mathrm{d}^{16}\mathrm{O}}{\mathrm{d}t} + m_{{}^{17}\mathrm{O}}\frac{\mathrm{d}^{17}\mathrm{O}}{\mathrm{d}t}\right],
\end{aligned}
\tag{6.294}
$$

la dernière équation exprime la conservation du nombre de baryons, elle permet de tenir compte des éléments qui, comme le ^{20}Ne, n'est brulé par aucune réaction, mais qui est crée, m_i est le nombre atomique de l'espèce chimique i.

Sans élément à l'équilibre, il faut remplacer les trois premières et la dernière équations par:

$$\begin{aligned}
\frac{\mathrm{d}^1\mathrm{H}}{\mathrm{d}t} &= -2R_1\ {}^1\mathrm{H}\ {}^2 - R_2\ {}^1\mathrm{H}\ {}^2\mathrm{H} + 2R_3\ {}^3\mathrm{He}^2 - R_5\ {}^1\mathrm{H}\ {}^7\mathrm{Li} - R_7\ {}^1\mathrm{H}\ {}^7\mathrm{Be} \\
&\quad -R_8\ {}^1\mathrm{H}\ {}^{12}\mathrm{C} - R_9\ {}^1\mathrm{H}\ {}^{13}\mathrm{C} - R_{10}\ {}^1\mathrm{H}\ {}^{14}\mathrm{N} \\
&\quad -(R_{11}+R_{12}){}^1\mathrm{H}\ {}^{15}\mathrm{N} - R_{13}\ {}^1\mathrm{H}\ {}^{16}\mathrm{O} - R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O}, \\
\frac{\mathrm{d}^2\mathrm{H}}{\mathrm{d}t} &= R_1\ {}^1\mathrm{H}\ {}^2 - R_2\ {}^1\mathrm{H}\ {}^2\mathrm{H}, \\
\frac{\mathrm{d}^3\mathrm{He}}{\mathrm{d}t} &= R_2\ {}^1\mathrm{H}\ {}^2\mathrm{H} - 2R_3\ {}^3\mathrm{He}^2 - R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He}, \\
\frac{\mathrm{d}^4\mathrm{He}}{\mathrm{d}t} &= R_3\ {}^3\mathrm{He}^2 - R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He} + 2R_5\ {}^1\mathrm{H}\ {}^7\mathrm{Li} + 2R_7\ {}^1\mathrm{H}\ {}^7\mathrm{Be} \\
&\quad +R_{12}\ {}^1\mathrm{H}\ {}^{15}\mathrm{N} + R_{14}\ {}^1\mathrm{H}\ {}^{17}\mathrm{O} \\
&\quad -3R_{15}\ {}^4\mathrm{He}^3 - R_{16}\ {}^4\mathrm{He}\ {}^{12}\mathrm{C} - R_{17}\ {}^4\mathrm{He}\ {}^{16}\mathrm{O}, \\
\frac{\mathrm{d}^7\mathrm{Li}}{\mathrm{d}t} &= -R_5\ {}^1\mathrm{H}\ {}^7\mathrm{Li} + R_6\ {}^7\mathrm{Be}\ e^-, \\
\frac{\mathrm{d}^7\mathrm{Be}}{\mathrm{d}t} &= R_4\ {}^3\mathrm{He}\ {}^4\mathrm{He} - R_6\ {}^7\mathrm{Be}\ e^- - R_7\ {}^1\mathrm{H}\ {}^7\mathrm{Be}. \\
\frac{\mathrm{d}\,\mathrm{Ex}}{\mathrm{d}t} &= -\frac{1}{m_{\mathrm{Ex}}}\left[m_{{}^1\mathrm{H}}\frac{\mathrm{d}^1\mathrm{H}}{\mathrm{d}t} + m_{{}^2\mathrm{H}}\frac{\mathrm{d}^2\mathrm{H}}{\mathrm{d}t} + m_{{}^3\mathrm{He}}\frac{\mathrm{d}^3\mathrm{He}}{\mathrm{d}t} + m_{{}^4\mathrm{He}}\frac{\mathrm{d}^4\mathrm{He}}{\mathrm{d}t} \right. \\
&\quad +m_{{}^7\mathrm{Li}}\frac{\mathrm{d}^7\mathrm{Li}}{\mathrm{d}t} + m_{{}^7\mathrm{Be}}\frac{\mathrm{d}^7\mathrm{Be}}{\mathrm{d}t} + m_{{}^{12}\mathrm{C}}\frac{\mathrm{d}^{12}\mathrm{C}}{\mathrm{d}t} + m_{{}^{13}\mathrm{C}}\frac{\mathrm{d}^{13}\mathrm{C}}{\mathrm{d}t} \\
&\quad \left. +m_{{}^{14}\mathrm{N}}\frac{\mathrm{d}^{14}\mathrm{N}}{\mathrm{d}t} + m_{{}^{15}\mathrm{N}}\frac{\mathrm{d}^{15}\mathrm{N}}{\mathrm{d}t} + m_{{}^{16}\mathrm{O}}\frac{\mathrm{d}^{16}\mathrm{O}}{\mathrm{d}t} + m_{{}^{17}\mathrm{O}}\frac{\mathrm{d}^{17}\mathrm{O}}{\mathrm{d}t} \right].
\end{aligned}$$

On dénommera par "jacobien" la matrice des dérivées partielles des équations d'évolution par rapport aux abondances:

$$J(i,j) \equiv \left(\frac{\partial \frac{\mathrm{d}x_i}{\mathrm{d}t}}{\partial x_j} \right)_{i,j}. \tag{6.295}$$

Cette matrice est la base de la résolution des équations implicites du problème différentiel raide, *cf.* § ?? (Page ??), constitué par le réseau nucléaire. Il est nécessaire d'en tenir compte, du fait que l'effet d'écran est fonction des abondances. Une erreur dans l'expression du jacobien entraîne des difficultés de convergence de l'algorithme d'intégration numérique des équations d'évolution. Le programme `test_nuc` du sous-directory `TESTS` peut servir de canevas pour construire un programme permettant de vérifier l'exactitude du jacobien *cf.* § 6.1.4 (Page 55).

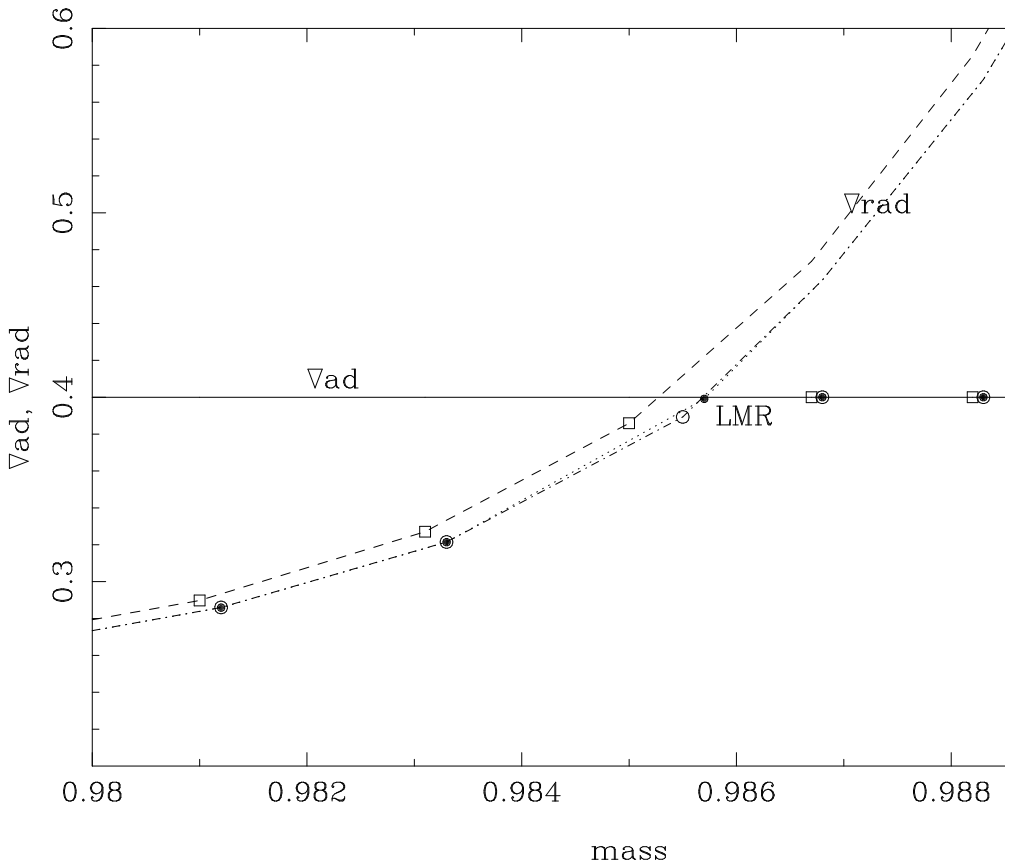


Figure 6.3: Two iterations were required to position a grid point on a RZ/CZ boundary. During the iterative process, the radiative gradient $\nabla_{\rm rad}$ decreased slightly, and the boundary moved to the right, while the adiabatic gradient $\nabla_{\rm ad}$ remained unchanged. The grid points successively moved from their initial positions, empty squares, to their final positions, empty circles and then full circles.

Chapter 7

Physical subroutines

Les grands diseux ne sont pas les grands faiseux.

Proverbe populaire.

Contents

The implementation of all these routines has benefited from numerous collaborations with, in particular, J.Provost, G.Berthomieu and B.Pichon.

We first mention the programming rules used in Cesam2k20, then describe the programming aspect of the physics routines grouped together in the modules of the `src` sub-directory. Most of these are simply applications to the problem of the internal structure of the numerical methods developed in Chapter 6. The description of purely numerical routines is the subject of Chapter 8.

For ease of use, the routines are described in the alphabetical order of their names.

7.1 Generic subroutines

The selected physics is introduced via the keywords in the data file `my_modele.don`, see Sect. **??**. Each keyword is associated with a **??**, which will be assigned the *dependent routine* where the calculation will be performed. For example, in Sect. 3.3 data file `ppcno9` is assigned to the `nom_nuc` keyword. This keyword associates the generic routine `nuc` with the function of the routine `ppcno9`. Thus, each time `call nuc( ... )` is called, thermonuclear reactions will be calculated by `ppcno9`, see Sect. sec:ppcno9.

7.2 Units

The arguments of the routines are in *cgs unit*, except: mass, luminosity and radius which are in solar units $M_\odot$,$R_\odot$ and $L_\odot$. If, in a sub-program, the evaluation of certain output variables involves input variables whose units are in different systems, the consistency between the units of the input and output variables is respected. For example, in the sub-programme `thermo` (see Sect. 7.80), which calculates various quantities and their first derivatives from thermodynamic variables, the variable `grad` ($\nabla = \partial \ln T / \partial \ln P$) is dimensionless. Calculating the variable `dgradm` ($\partial\nabla/\partial m$) requires the input variable `m` (mass) which is dimensionless: $[m/M_\odot]$ so, `dgradm` is dimensionless and not g^{-1}. Variable names are chosen to be easily identifiable.

The chemical composition is an input argument for various subroutines. The abundances per mole (resp. per gram per atomic mass) of the various elements are the values of a B-Spline very often called: `chim` where the isotopes *actually used* are often arranged in order of increasing atomic masses, an order which is defined in the sub-programme `tabul_nuc` (see Sect. 7.77), when creating nuclear reaction tables (see Sect 6.1.4). This order is not imperative, except for ^{1}H which must be the first isotope.

Some variables are used to store the index of specific elements:

- Index of ^{1}H: `1`.
- Index of ^{4}He: `ihe4`.
- Index of ^{3}He: `ihe4-1`.
- Index of ^{56}Fe: `ife56`.
- Index of the supplementary ficticious element, see Sect. 6.10.1.

All variables have an *explicit* type, except for certain programs from an external source, for example: `z14xcotrin21` (see Sect. 7.85). Most of the fundamental physical constants, and variables fixed once and for all used by Cesam2k20 are part of the module `mod_donnees` (see Sect. D.3). Fundamental constants are initialised at the beginning of each execution by a call to the sub-programme `ini_ctes` (see Sect. 7.35), at the beginning of the sub-programme `cesam` (see Sect. 7.7.3). The constants specific

to each routine are initialised or, if possible, recalculated using the fundamental constants at the first call of this routine, and are preserved by using the instruction save.

Comments and bibliographical references are used to monitor and check the algorithms. In each sub-program, the quantities entering and leaving the argument list are listed, sometimes rather succinctly. In a sub-program, variables and routines external to the module to which the sub-program belongs are introduced by the instruction:

```
use nom_module, only : routine, variable
```

This constraint allows the identification of the modules to which the invoked quantities belong.

As the number of layers and the limits vary with time, extensive use is made of allocatable arrays which are allocated/deallocated as required.

The notations used for the variables are as explicit as possible, bearing in mind the need for conciseness (e.g., we use gradad for the adiabatic gradient $\nabla_{\rm ad}$). Similarly for the derivatives (e.g, the derivatives with respect to $P_{\rm gaz} \equiv$ p have a prefix d and an extension[1] p, for example: dgradadp$\equiv \frac{\partial \nabla_{\rm ad}}{\partial P_{\rm gaz}}$).

7.3 Module mod_kind

Defines floating-point precision

Members:

dp : **integer**, **parameter**, **public**
Real double precision.

sp : **integer**, **parameter**, **public**
Real single precision.

dpc : **integer**, **parameter**, **public**
Complex double precision.

7.4 Module mod_communicate

Module dedicated to handling commmunication with user. Currently only supports error-handling.

Generic type that can hold any type of variable.

This type is used when passing a variable in an error message.

object: **type**, **public**

Members:

item : **class**(*), **pointer**
Variable of any type. Initial value: item=>null().

fmt : **character**(len=8)
Format to write this variable.

inited : **logical**
True if type inited. Initial value: inited=.false..

pr : **type**(ptrr)
Pointer to real expression.

pi : **type**(ptri)
Pointer to integer expression.

[1] This rule is often broken for reasons of conciseness.

Type used to communicate with user. Currently implemented: raise error message
communicator: **type**, **public**

Members:

values : **type**(object), **allocatable**, **dimension**(:)
Values that will be formated in messages.

fr_fmt : **character**(len=1200)
French format (10 lines).

en_fmt : **character**(len=1200)
English format (10 lines).

language : **character**(len=10)
Wrpper of params%langue.

message : **character**(len=1200)
Unidentified language format (10 lines).

error : **character**(len=1200)
Unidentified language format (10 lines).

parent : **integer**
Integer associated to the module where the message or the error has been raised.

errID : **integer**
ID of the error to be raised.

fmtID : **integer**
ID of the format to be used.

routine : **character**(len=40)
Name of the routine where the message or the error has been raised.

werror : **logical**
If True, stop Cesam2k20. Initial value: werror=.false..

warning : **logical**
If True, requires attention. Initial value: warning=.false..

nl : **integer**
Number of lines in the format.

nv : **integer**
Number of values in the format. Initial value: nv=0.

inited : **logical**
If true, the communicator has been inited. Initial value: inited=.false..

seperate_output : **logical**
Wrapper of mod_donnees.Baratine. Initial value: seperate_output=.false..

7.4.1 Sub-module submod_errors

7.5 Module mod_donnees

7.5.1 Type constants

constants: **type**

Members:

amu : **real**(kind=dp), **public**
Atomic mass unity (Avogadro number is 1/amu.

aradia : **real**(kind=dp), **public**
Adiation constant.

clight : **real**(kind=dp), **public**
Celerity of light.

echarg : **real**(kind=dp), **public**
Electronic charge.

eve : **real**(kind=dp), **public**
Electron volt.

gmsol : **real**(kind=dp), **public**
Product $\mathcal{G}M_\odot$.

hpl : **real**(kind=dp), **public**
Planck's constant.

kbol : **real**(kind=dp), **public**
Boltzman's constant.

granr : **real**(kind=dp), **public**
Ideal gas constant.

ln10 : **real**(kind=dp), **public**
$\ln 10$.

lbol0 : **real**(kind=dp), **public**
0 point of the bolometric magnitude $M_{\rm bol}$.

me : **real**(kind=dp), **public**
Electron's mass.

pi : **real**(kind=dp), **public**
π.

secon6 : **real**(kind=dp), **public**
Number of seconds in 1 Myr (1967's norm).

sigma : **real**(kind=dp), **public**
Stefan's constant.

zsx_sol : **real**(kind=dp), **public**
$Z/X_\odot$.

fesh_sol : **real**(kind=dp), **public**
$[{\rm Fe/H}]_\odot$.

ua : **real**(kind=dp), **public**
Astronomical unit.

lsol : **real**(kind=dp), **public**
Solar luminosity.

msol : **real**(kind=dp), **public**
Solar mass.

rsol : **real**(kind=dp), **public**
Solar radius.

mterre : **real**(kind=dp), **public**
Terrestrial mass.

g : **real**(kind=dp), **public**
Gravity.

an : **real**(kind=dp), **public**
Atomic mass of neutron in unit of amu.

ap : **real**(kind=dp), **public**
Atomic mass of proton in unit of amu.

ah : **real**(kind=dp), **public**
Atomic mass of l'hydrogène in unit of amu.

ah2 : **real**(kind=dp), **public**
Atomic mass of deutérium in unit of amu.

ahe3 : **real**(kind=dp), **public**
Atomic mass of l'hélium 3 in unit of amu.

ahe4 : **real**(kind=dp), **public**
Atomic mass of l'hélium 4 in unit of amu.

ali6 : **real**(kind=dp), **public**
Atomic mass of lithium 6 in unit of amu.

ali7 : **real**(kind=dp), **public**
Atomic mass of lithium 7 in unit of amu.

abe7 : **real**(kind=dp), **public**
Atomic mass of béryllium 7 in unit of amu.

abe9 : **real**(kind=dp), **public**
Atomic mass of béryllium 9 in unit of amu.

ab11 : **real**(kind=dp), **public**
Atomic mass of bore 11 in unit of amu.

ac12 : **real**(kind=dp), **public**
Atomic mass of carbone 12 in unit of amu.

ac13 : **real**(kind=dp), **public**
Atomic mass of carbone 13 in unit of amu.

an13 : **real**(kind=dp), **public**
Atomic mass of l'azote 13 in unit of amu.

an14 : **real**(kind=dp), **public**
Atomic mass of l'azote 14 in unit of amu.

an15 : **real**(kind=dp), **public**
Atomic mass of l'azote 15 in unit of amu.

ao16 : **real**(kind=dp), **public**
Atomic mass of l'oxygène 16 in unit of amu.

ao17 : **real**(kind=dp), **public**
Atomic mass of l'oxygène 17 in unit of amu.

ao18 : **real**(kind=dp), **public**
Atomic mass of l'oxygène 18 in unit of amu.

afe56 : **real**(kind=dp), **public**
Atomic mass of fer 56 in unit of amu.

af18 : **real**(kind=dp), **public**
Atomic mass of fluor 18 in unit of amu.

af19 : **real**(kind=dp), **public**
Atomic mass of fluor 19 in unit of amu.

ane20 : **real**(kind=dp), **public**
Atomic mass of néon 20 in unit of amu.

ane21 : **real**(kind=dp), **public**
Atomic mass of néon 21 in unit of amu.

ane22 : **real**(kind=dp), **public**
Atomic mass of néon 22 in unit of amu.

ana23 : **real**(kind=dp), **public**
Atomic mass of sodium 23 in unit of amu.

amg23 : **real**(kind=dp), **public**
Atomic mass of magnésium 23 in unit of amu.

amg24 : **real**(kind=dp), **public**
Atomic mass of magnésium 24 in unit of amu.

amg25 : **real**(kind=dp), **public**
Atomic mass of magnésium 25 in unit of amu.

amg26 : **real**(kind=dp), **public**
Atomic mass of magnésium 26 in unit of amu.

asi28 : **real**(kind=dp), **public**
Atomic mass of silicium 28 in unit of amu.

aal27 : **real**(kind=dp), **public**
Atomic mass of l'aluminium 27 in unit of amu.

as32 : **real**(kind=dp), **public**
Atomic mass of soufre 32 in unit of amu.

ap31 : **real**(kind=dp), **public**
Atomic mass of phosphore 31 in unit of amu.

aca40 : **real**(kind=dp), **public**
Atomic mass of calcium 40 in unit of amu.

ani58 : **real**(kind=dp), **public**
Atomic mass of nickel 58 in unit of amu.

delta_nu_sol : **real**(kind=dp), **public**
Solar large separation.

nu_max_sol : **real**(kind=dp), **public**
Solar nu max.

vmax_sol : **real**(kind=dp), **public**
Solar radial mode velocity at $\nu_{\max}$.

w_sol : **real**(kind=dp), **public**
Solar angular velocity of Carrington.

inited : **logical**
True if the type is inited. Initial value: inited =.false..

source : **character** (len=50)
Bibliographical source where the values of the constants are taken.

Subroutine ctes_85

```
subroutine ctes_85( this )
```

Private routine of module mod_donnees.
Routine that initialize physical constants.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

GONG : Number of seconds in one year.

Guenter : Solar values from Guenter & al.

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

GN93 : Solar [Fe/H] from (Grevesse & Noels, 1993).

Orignal author(s):

- P.Morel, Departement J.D. Cassini : O.C.A., CESAM2k.
- J. Marques: transition to class(constants).

Subroutine ctes_94

```
subroutine ctes_94( this )
```

Private routine of module mod_donnees.
Routine that initialize physical constants.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

Guenter : Solar values from Guenter & al.

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

GN93 : Solar [Fe/H] from (Grevesse & Noels, 1993).

Orignal author(s):

- P.Morel, Departement J.D. Cassini : O.C.A., CESAM2k.
- J. Marques: transition to class(constants).

Subroutine ctes_94m

```
subroutine ctes_94m( this )
```

Private routine of module mod_donnees.
Routine that initialize physical constants. Identical to ctes_94 but atomic masses in unit of amu are set to nearest integer value.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

Guenter : Solar values from Guenter & al.

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

GN93 : Solar [Fe/H] from (Grevesse & Noels, 1993).

Orignal author(s):

- P.Morel, Departement J.D. Cassini : O.C.A., CESAM2k.
- J. Marques: transition to class(constants).

Subroutine ctes_94_asplund

```
subroutine ctes_94_asplund( this )
```

Private routine of module mod_donnees.
Routine that initialize physical constants. Identical to ctes_94 but atomic masses in unit of amu are set to nearest integer value.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

Guenter : Solar values from Guenter & al.

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

AGSS09 : Solar [Fe/H] from (Asplund et al., 2009).

Orignal author(s):

- Y. Lebreton, GEPI, obs. Paris.
- J. Marques: transition to class(constants).

Subroutine ctes_gs98

```
subroutine ctes_gs98( this )
```

Private routine of module mod_donnees.

Routine that initialize physical constants. Extra parameter extra\%source_cts allows to come back to original values if set to 'Lide1994'. extra\%source_cts = 'IAU2015_CODATA2018' constants are in accordance with ctes_aag21.f90.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

IAU2015 : CODATA2018: Physical constants taken from PLATO WP120 - Constants (CODATA 2018).

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

GS98 : Solar [Fe/H] from (Grevesse & Sauval, 1998).

Orignal author(s):

- Y. Lebreton, LESIA, Obs. Paris.
- J. Marques: transition to class(constants).

Subroutine ctes_aag21

```
subroutine ctes_aag21( this )
```

Private routine of module mod_donnees.

Routine that initialize physical constants. Extra parameter extra\%source_cts allows to come back to original values if set to 'Lide1994'. extra\%source_cts = 'IAU2015_CODATA2018' constants are in accordance with ctes_aag21.f90.

Arguments:

this : **class**(constants), **intent**(inout)

References:

Lide1994 : Physical constants from Lide (1994).

IAU2015 : CODATA2018: Physical constants taken from PLATO WP120 - Constants (CODATA 2018).

NACRE : Atomic masses in units of amu are taken from NACRE compilation.

AAG21 : Solar [Fe/H] from (Asplund et al., 2021).

Orignal author(s):

- L. Manchon, MPS, Gottingen.
- Y. Lebreton, LESIA, Obs. Paris.
- J. Marques: transition to class(constants).

7.5.2 Type numerical_parameters

numerical_parameters: type, public

Members:

ini0 : integer
Number of Newton-Raphson iterations with re-estimation of chemical composition, and limits RZ/CZ. Initial value: ini0=4.

l0 : integer
DepreciatedIn output ASCII files, number of points added in the neighbourhood of limits ZR/CZ. Initial value: l0=0.

m_qs : integer
Order of the B-Splines for interpolation of quasi-static variables. Initial value: m_qs=2.

m_ch : integer
Order of the B-Splines for interpolation of the chemical composition. Initial value: m_ch=2.

m_vth : integer
Order of the B-Splines for interpolation of thermodynamic variblaes (only used in rep.Osc_v3.f90). Initial value: m_vth=3.

m_rot : integer
Order of the B-Splines for interpolation of rotation quantities. Initial value: m_rot=3.

m_tds : integer
Order of the B-Splines for interpolation of TdS. Initial value: m_tds=2.

m_ptm : integer
Ordre of the B-Splines for interpolation of mass (used when there is mass loss. Initial value: m_ptm=2.

n_atm : integer
Number of layers in the atmosphere. Initial value: n_atm=75.

n_max : integer
Maximum number of layers. Initial value: n_max=2500.

n_min : integer
Minimum number of layers. Initial value: n_min=150.

ordre : integer
Order of the scheme used for the interpolation of nuclear reaction rates with rk_imps. Initial value: ordre=2.

yld_rep : integer
YLD : write one .Rep file every yld_rep (needs nbmax_modeles<0, all_rep=true). Initial value: yld_rep=1.

yld_osc : integer
YLD : write one .Osc file every yld_osc (if yld_osc<0, interface file with LOSC is writen). Initial value: yld_osc=1.

osc2d_step : integer
!> write one .Osc2d file every osc2d_step. Initial value: osc2d_step=5.

osc_step : integer
Write one .Osc file every osc_step (redundant with yld_osc). Initial value: osc_step=1.

m_qs2d : integer
Order of the spline for 2D interpolation of 1D quantities on 2D grid. Initial value: m_qs2d=3.

m_rot2d : **integer**
Order of the spline for 2D interpolation of rotation. Initial value: m_rot2d=3.

m_legendre : **integer**
Order of the decomposition on Legendre polynomials. Initial value: m_legendre=2.

n_theta : **integer**
Number of points in the angular mesh. Initial value: n_theta=100.

iter_qs : **integer**, **dimension**(7)
Controls quasi-static variables. Initial value: iter_qs=(/0,0,0,0,0,0,0/).

ctel : **real**(kind=dp)
Repartition constant for the luminosity. Initial value: ctel=0.0d0.

ctep : **real**(kind=dp)
Repartition constant for the pressure. Initial value: ctep=-1.0d0.

ctem : **real**(kind=dp)
Repartition constant for the mass. Initial value: ctem=15.0d0.

cter : **real**(kind=dp)
Repartition constant for the radius. Initial value: cter=0.0d0.

ctet : **real**(kind=dp)
Repartition constant for the temperature. Initial value: ctet=-1.0d0.

d_grav : **real**(kind=dp)
Maximum variation of $T\mathrm{d}S$. Initial value: d_grav=5.0d3.

d_grav_int : **real**(kind=dp)
Maximum variation of $T\mathrm{d}S$ inside static. Initial value: d_grav_int=5.0d3.

dlntc : **real**(kind=dp)
Maximum variation of core temperature between two consecutive time steps. Initial value: dlntc=0.07d0.

drhoc : **real**(kind=dp)
Maximum relative variation of cenral density between two consecutive time steps. Initial value: drhoc=-1.0_dp.

dteff : **real**(kind=dp)
Maximum relative variation of T_{eff} between two consecutive time steps. Initial value: dteff=-1.0_dp.

dlogg : **real**(kind=dp)
Maximum dex variation of $\log g$ between two consecutive time steps. Initial value: dlogg=-1.0_dp.

dalpha : **real**(kind=dp)
Maximum relative variation of α between two consecutive time steps. Initial value: dalpha=0.03_dp.

dlum : **real**(kind=dp)
Maximum dex variation of luminosity between two consecutive time steps. Initial value: dlum=-1.0_dp.

dw : **real**(kind=dp)
Maximum relative variation of central Ω between two consecutive time steps. Initial value: dw=-1.0_dp.

dsenv : **real**(kind=dp)
Maximum variation of adiabatic entropy between two consecutive time steps. Initial value: dsenv=5.0d-4.

evolved : **real**(kind=dp)
Fraction of the initial X content below which variations of $T_{\rm eff}$, luminosity, $\log g$ and $\rho_{\rm c}$ are limited. Initial value: `evolved=0.1_dp`.

dn_fixe : **real**(kind=dp)
Modification rate for a fixed grid of chemical composition. Initial value: `dn_fixe=0.05d0`.

dpsi : **real**(kind=dp)
Maximum variation of ψ between two consecutive time steps before `n_qs` is modify. Initial value: `dpsi=0.05d0`.

dt0 : **real**(kind=dp)
Initial time step when starting from ZAMS. Initial value: `dt0=1.0d0`.

dtmax : **real**(kind=dp)
Maximum time step. Initial value: `dtmax=200.0d0`.

dtmin : **real**(kind=dp)
Minimum time step. Initial value: `dtmin=1.0d-15`.

fmin_abon : **real**(kind=dp)
Factor linking `ab_min` et `abon_ini`. Initial value: `fmin_abon=0.05d0`.

loc_zc : **real**(kind=dp)
Precision of locating RC/CZ boundaries. Initial value: `loc_zc=1.0d-3`.

precix : **real**(kind=dp)
Precision on Newton-Raphson iterations for spatial integrations. Initial value: `precix=1.0d-3`.

precit : **real**(kind=dp)
Maximum relative variation for temporal integration of chemical composition. Initial value: `precit=0.15d0`.

psi0 : **real**(kind=dp)
Distribution constant to ensure. Initial value: `psi0=0.08d0`.

ro_test : **real**(kind=dp)
Test for variation of gravitational energy if `rho > ro_test`. Initial value: `ro_test=0.1d0`.

q0 : **real**(kind=dp)
In output ASCII files, a point is placed at $q0 > 0$ times the spacing between the first two points. Initial value: `q0=0.05d0`.

yld_dx : **real**(kind=dp)
YLD : Maximum relative variation of X in the core between two consecutive time steps. Initial value: `yld_dx=-1.0d0`.

yld_rac : **real**(kind=dp)
YLD : Optical depth of the matching between interior and atmosphere. Initial value: `yld_rac=1.0d0`.

tau_min : **real**(kind=dp)
YLD : Minimum optical depth. Initial value: `tau_min=1.0d-4`.

dt_m : **real**(kind=dp)
Parameter needed to modulate the value of `dt_max` with the value of the mass: `dtmax = dtmax / mtot**(dt_m)`. Initial value: `dt_m=-2.5_dp`.

x_tams : **real**(kind=dp)
Fraction of hydrogen at center that defines TAMS. Initial value: `x_tams=0.01d0`.

y_agb : **real**(kind=dp)
Fraction of helium at center that defines AGB. Initial value: `y_agb=0.001d0`.

dx_tams : **real**(kind=dp)
Precision on the fraction of hydrogen at center that defines TAMS. Initial value: dx_tams=1.0d-4.

mcz_ext_min : **real**(kind=dp)
External CZ must have at least this fraction of mstar. Initial value: mcz_ext_min=1.0d-10.

exact_stop : **logical**
If fast_rot > 0.0, the first fast_rot Myrs are computed with a slow rotation rate. The rotation is increased linearly with age until it reaches the desired initial rotation rate at an age of fast_rot. This avoids the case of breakup velocity reached at the beginning. [[Should be moved to type_extra]] If true, a last model is computed to match exactly the stop criteria (e.g. To reach given core temperature). Initial value: exact_stop=.True..

en_masse : **logical**
If true, use lagrangian coordinates. Initial value: en_masse=.True..

kipp : **logical**
If true, use Kippenhahn's approximation $T\mathrm{d}S = \mathrm{d}E + P\mathrm{d}V$. Initial value: kipp=.false..

lisse : **logical**
If true, chemical composition and rotation quantities are smoothed. Initial value: lisse=.false..

mu_saha : **logical**
If true, we use Saha equation to compute mean molecular weight μ. Initial value: mu_saha=.True..

mvt_dis : **logical**
If true, adjustment of chemical composition due to discontinuity displacements. Initial value: mvt_dis=.True..

new_bv : **logical**
If true, calculate the Brunt-Vaissala freq using $\varphi = \frac{\mathrm{d}\ln\rho}{\mathrm{d}\ln\mu}$. Initial value: new_bv=.false..

yld_hr : **logical**
YLD : If true, writes a special file .HRYL. Initial value: yld_hr=.false..

yld_l23 : **logical**
YLD : If true, the unknown is $l^{2/3}$ instead of l. Initial value: yld_l23=.false..

yld_fgong : **logical**
YLD : If true, fgong files are generated. Initial value: yld_fgong=.false..

d2 : **logical**
If true, we compute the 2D structure of the star according to Roxburgh 2006 method. Initial value: d2=.True..

general : **logical**
If True, general structure equation are used. Initial value: general=.false..

simple : **logical**
If true, Omega_2 is not accounted for. Initial value: simple=.True..

extended_osc2d : **logical**
If true, more quantities are added to osc2d files. Initial value: extended_osc2d=.false..

vdiff_out : **logical**
MD : If true, write diffusion velocities to an ASCII file. Initial value: vdiff_out=.false..

grad_out : **logical**
MD : If true, write radiative accelerations to an ASCII file. Initial value: grad_out=.false..

th_out : **logical**
MD : If true, write information about thermohaline convection to an ASCII file. Initial value: th_out=.false..

eos_inplace : **logical**
If true, EoS table must be stored in the directory when the model is computed. Initial value: eos_inplace=.false..

ec : **logical**
If true, an entropy calibration is performed. Initial value: ec=.false..

out_pms : **logical**
Output.Osc on PMS. Initial value: out_pms=.True..

clean : **logical**
Output.Osc on PMS. Initial value: clean=.True..

precision : **character**(len=2)
Default precision. Initial value: precision='co'.

ltau_rep : **character**(len=3)
$\log(\tau)$ follows either a linear trend or a piecewise cubic trend. Initial value: ltau_rep='cub'.

ec_s : **character**(len=7)
Method of determination of adiabatic entropy (available options 'bottom' or 'average'). Initial value: ec_s='average'.

lim_zc : **character**(len=9)
Switch between original lim_zc, and lim_zc that follows Gabriel (2014)'s recommendations. Initial value: lim_zc='gabriel14'.

inited : **logical**
True if the type is correctly initialized. Initial value: inited=.false..

7.6 Module mod_nuc

7.6.1 Subroutine abon_ini

```
subroutine abon_ini( )
```

Private routine of mod_nuc initialisation of abundances, isotope ratios and element charges. The initial abundances of the isotopes used in the nuclear network will be deduced from these initial values depending on the chain of reactions used and the isotopes diffused. Source : ftp://www-phys.Llnl.Gov/pub/opal

To add an element: In mod_nuc module, increase the parameter nelem_ini of 1.
eps_ini added by Y. Lebreton for Irwin's EOS.

Orignal author(s):

- P. Morel Département Cassiopée, O.C.A.

Advisor(s):

- F. Thévenin Département Cassiopée, O.C.A.

The main function of abon_ini is to initialise the abundances, isotope ratios and charges of the chemical elements used by the model. Following this initialization in the tabul_nuc, see Fig. 7.3 (Page 209) routine, only the chemical elements used in thermonuclear reactions will be retained. The source of the data is mainly the OPAL website: ftp://www-phys.llnl.gov/pub/opal.

The charges, atomic masses and symbols of the chemical elements up to nickel ($Z = 28$) are initialized, followed by a certain number of isotopic ratios [2] routine. Depending on the initial mixture chosen, the abundances in dex (H=12) or in number are initialised and then, if necessary, transformed into abundances by number. In cases where an initial personalised mixture is used, this is read, in numbers, either in the file `mon_modele.mix`, or in the file `xmixture` (see § 3.15 (Page 30)). The relative abundances of the metals in Z are then evaluated.

`abon_ini` is called by the nuclear reaction calculation routine when it is initialised (see *cf.* § 7.6.1 (Page 149)).

Subroutine nuc

```
subroutine nuc( t, ro, comp, dcomp, jac, deriv, fait, epsilon, et, ero, ex, hhe, be7e,
↪  b8e, n13e, o15e, f17e, eps_all_reacs )
```

Public routine of `mod_nuc`.
Generic routine that compute thermonuclear reaction rates.

Arguments:

t : **real**(kind=dp), **intent**(in)
Temperature.

ro : **real**(kind=dp), **intent**(in)
Density.

fait : **integer**, **intent**(in)
`fait=1` : Initialize chemical composition
`fait=2` : lst derivative and jacobian if deriv is True
`fait=3` : nuclear energy and time derivatives
`fait=4` : neutrino production.

deriv : **logical**, **intent**(in)
If true: computes jacobian.

comp : **real**(kind=dp), **intent**(inout), **dimension**(:)
Abundances.

jac : **real**(kind=dp), **intent**(out), **dimension**(:,:)
Jacobian (in units of Myrs).

dcomp : **real**(kind=dp), **intent**(out), **dimension**(:)
Time derivatives of abundances (in units of Myrs).

epsilon : **real**(kind=dp), **intent**(out), **dimension**(:)
Thermonuclear energies.

ex : **real**(kind=dp), **intent**(out), **dimension**(:)
X-derivative of thermonuclear energies.

et : **real**(kind=dp), **intent**(out)
Temperature-derivative of thermonuclear energies.

ero : **real**(kind=dp), **intent**(out)
Density-derivative of thermonuclear energies.

hhe : **real**(kind=dp), **intent**(out)
Number of neutrino for H + He [g/s^1].

[2] These initializations would be more justified in the `ctes`.

be7e : **real**(kind=dp), **intent**(out)
Number of neutrino for $Be_7 + e^+$ [g/s^1].

b8e : **real**(kind=dp), **intent**(out)
Number of neutrino for $B_8 + e^+$ [g/s^1].

n13e : **real**(kind=dp), **intent**(out)
Number of neutrino for $N_{13} + e^+$ [g/s^1].

o15e : **real**(kind=dp), **intent**(out)
Number of neutrino for $O_{15} + e^+$ [g/s^1].

f17e : **real**(kind=dp), **intent**(out)
Number of neutrino for $F_{17} + e^+$ [g/s^1].

eps_all_reacs : **real**(kind=dp), **optional**, **intent**(out), **dimension**(:)
Nuclear reaction rates of all reactions seperatly (not fully implemented).

Orignal author(s):

- P. Morel, Département J.D. Cassini, O.C.A.

The nuc type routines are available in four different categories, depending on the temperature range covered:

test: simplified reaction networks for test purposes.

pre-3α: usable from the PMS to the initiation of helium reactions, temperature range covered: 0.5-100 MK. The reaction networks used describe the PP and CNO cycles.

3α: usable from the PMS to the initiation of carbon reactions, temperature range covered: 0.5-200 MK. The reaction networks used describe the PP, CNO and 3α cycles.

co: usable from the PMS to the oxygen cycle, temperature range covered: 0.5-2000 MK. The reaction networks used describe the PP, CNO, 3α cycles, and the combustion of carbon and oxygen. Beyond this, the quasi-static equilibrium assumption is no longer justified.

The isotopes used in the nuclear networks depend, on the one hand, on the type of model to be calculated and, on the other hand, on the chemical elements of particular interest. The implementation of these routines has benefited in various ways from the collaboration of B. Pichon, Y. Lebreton, S. Brun and D. Cordier. The Cesam2k20 source contains various routines of the type nuc specially adapted to specific cases of stellar evolution.

We describe the flowchart of a routine of type nuc, with the exception of iben, *cf.* §7.34 (Page 201), and pp1, *cf.* §7.6.1 (Page 151), whose algorithms differ somewhat from the general framework. An input argument, fait, defines the part of the calculation to be performed :

- fait=1
  - Initialisation of the number of isotopes in the nuclear network.
  - Initialisation of the nuclear network by calling rqreac, *cf.* §7.72 (Page 224).
  - Initialisation of abundances by calling abon_ini, *cf.* §7.6.1 (Page 148).
  - Determination of the type and characteristics of the fictitious isotope and calculation of the initial isotopic abundances.
  - Various initialisations and writings.
- fait=2
  - Calculation of reaction rates.

- Formation of the Jacobian.

- `fait=3`
  - Calculation of nuclear energy.

- `fait=4`
  - Calculation of neutrino fluxes. The "fluxes to earth" are calculated assuming that these neutrinos are produced within 1 AU of the earth, even though the model is not a solar model.

`nuc` is called from various places, in particular from `rk_imps`, `thermo`, `eq_dif_chim`, `cesam`.

Subroutine pp1

```
subroutine pp1(t, ro, comp, dcomp, jac, deriv, fait, epsilon, et, ero, ex, hhe, be7e,
↪  b8e, n13e, o15e, f17e, eps_all_reacs)
```

Private routine of `mod_nuc`.
Thermonuclear reactions from GONG.
Auteur: P. Morel, Département J.D. Cassini, O.C.A.
According to the *"Computational procedures for GONG solar model project"* (Christensen-Dalsgaard, 1988), only the isotope ^{1}H is explicitly taken into account. The initial hydrogen abundance, reaction rate and energetics are calculated *in situ* by an analytical approximation of the PP cycle.

Subroutine pp3

```
subroutine pp3( t, ro, comp, dcomp, jac, deriv, fait, epsilon, et, ero, ex, hhe,be7e,
↪  b8e, n13e, o15e, f17e, eps_all_reacs )
```

Private routine of `mod_nuc`.
`pp3` was made essentially for tests purposes.
PP cycle with elements: ^{1}H (index 1), ^{3}He (index 2), ^{4}He (index 3), Ex (index 4). Ex is a complementary fictive element, only used for diffusion. ^{2}H, ^{7}Li and ^{7}Be are considered to be at equilibrium.
At first call, the reaction table is automatically created by `rq_reac` and `tabul_nuc`.
Initialise `ab_min` (negligeable abondances) and `ab_ini` (initial abondances).
`r(1)` : reaction $^1\mathrm{H}(p, e^+\nu)^2\mathrm{H}$
`r(2)` : reaction $^2\mathrm{H}(p, g)^3\mathrm{H}$
`r(3)` : reaction $^3\mathrm{He}(^3\mathrm{He}, 2p)^4\mathrm{He}$
`r(4)` : reaction $^3\mathrm{He}(\alpha, g)^7\mathrm{Be}$
`r(5)` : reaction $^7\mathrm{Li}(p, \alpha)^4\mathrm{He}$
`r(6)` : reaction $^7\mathrm{Be}(e^-, \nu g)\mathrm{Li7}$
`r(7)` : reaction $^7\mathrm{Be}(p, g)^8\mathrm{B}(, e^+\nu)^8\mathrm{Be}(\alpha)^4\mathrm{He}$

References:

Clayton68 : Clayton, Principles of stellar evolution and nucleosynthesis, 1968, p. 380, 392 and 430.

Orignal author(s):

- P.Morel, Departement J.D. Cassiopee, O.C.A.

Subroutine ppcno9

```
subroutine ppcno9( t, ro, comp, dcomp, jac, deriv, fait, epsilon, et, ero, ex,hhe,
↪  be7e, b8e, n13e, o15e, f17e, eps_all_reacs )
```

Private routine of mod_nuc.

PP, CNO cycles with elements : ^{1}H (index 1), ^{3}He (index 2), ^{4}He (index 3), ^{12}C (index 4), ^{13}C (index 5), ^{14}N (index 6), ^{15}N (index 7), ^{16}O (index 8), ^{17}O (index 9), Ex (index 10). ^{2}H, ^{7}Li, ^{7}Be are considered at equilibrium.

A first call to rq_reac initialise the number of chemical elements for which the nuclear reactions are tabulated in ppcno9. To them, we add Ex, so we have soit nchim+1 elements

Initialise ab_min (negligeable abondances) and ab_ini (initial abondances).

PP chain:

r(1) : reaction $^{1}\mathrm{H}(p,e^{+},\nu)^{2}\mathrm{H}$
r(2) : reaction $^{2}\mathrm{H}(p,g)^{3}\mathrm{He}$
r(3) : reaction $^{3}\mathrm{He}(^{3}\mathrm{He},2p)^{4}\mathrm{He}$
r(4) : reaction $^{3}\mathrm{He}(\alpha,g)^{7}\mathrm{Be}$
r(5) : reaction $^{7}\mathrm{Li}(p,\alpha)^{4}\mathrm{He}$
r(6) : reaction $^{7}\mathrm{Be}(e^{-},\nu g)^{7}\mathrm{Li}$
r(7) : reaction $^{7}\mathrm{Be}(p,g)^{8}\mathrm{B}(e^{+},\nu)^{8}\mathrm{Be}(\alpha)^{4}\mathrm{He}$

CNO cycle:

r(8) : reaction $^{12}\mathrm{C}(p,g)^{13}\mathrm{N}(e^{+},\nu)^{13}\mathrm{C}$
r(9) : reaction $^{13}\mathrm{C}(p,g)^{14}\mathrm{N}$
r(10) : reaction $^{14}\mathrm{N}(p,g)^{15}\mathrm{O}(e^{+},nu)^{15}\mathrm{N}$
r(11) : reaction $^{15}\mathrm{N}(p,g)^{16}\mathrm{O}$
r(12) : reaction $^{15}\mathrm{N}(p,\alpha)^{12}\mathrm{C}$
r(13) : reaction $^{16}\mathrm{O}(p,g)^{17}\mathrm{F}(e^{+},\nu)^{17}\mathrm{O}$
r(14) : reaction $^{17}\mathrm{O}(p,\alpha)^{14}\mathrm{N}$

References:

Clayton68 : Clayton, Principles of stellar evolution and nucleosynthesis, 1968, p. 380, 392 and 430.

Orignal author(s):

- P.Morel, Département J.D. Cassini, O.C.A.

Subroutine ppcno10

```
subroutine ppcno10( t, ro, comp, dcomp, jac, deriv, fait,epsilon, et, ero, ex, hhe,
↪  be7e, b8e, n13e, o15e, f17e, eps_all_reacs )
```

Private routine of mod_nuc.

PP, CNO cycles with elements : ^{1}H (index 1), ^{3}He (index 2), ^{4}He (index 3), ^{7}Li (index 4), ^{12}C (index 5), ^{13}C (index 6), ^{14}N (index 7), ^{15}N (index 8), ^{16}O (index 9), ^{17}O (index 10), Ex (index 11). ^{2}H, ^{7}Li, ^{7}Be are considered at equilibrium.

A first call to rq_reac initialise the number of chemical elements for which the nuclear reactions are tabulated in ppcno10. To them, we add Ex, so we have soit nchim+1 elements

Initialise ab_min (negligeable abondances) and ab_ini (initial abondances).

PP chain:
r(1) : reaction $^{1}\mathrm{H}(p,e^{+},\nu)^{2}\mathrm{H}$
r(2) : reaction $^{2}\mathrm{H}(p,g)^{3}\mathrm{He}$
r(3) : reaction $^{3}\mathrm{He}(^{3}\mathrm{He},2p)^{4}\mathrm{He}$
r(4) : reaction $^{3}\mathrm{He}(\alpha,g)^{7}\mathrm{Be}$
r(5) : reaction $^{7}\mathrm{Li}(p,\alpha)^{4}\mathrm{He}$
r(6) : reaction $^{7}\mathrm{Be}(e^{-},\nu g)^{7}\mathrm{Li}$
r(7) : reaction $^{7}\mathrm{Be}(p,g)^{8}\mathrm{B}(e^{+},\nu)^{8}\mathrm{Be}(\alpha)^{4}\mathrm{He}$
CNO cycle:
r(8) : reaction $^{12}\mathrm{C}(p,g)^{13}\mathrm{N}(e^{+},\nu)^{13}\mathrm{C}$
r(9) : reaction $^{13}\mathrm{C}(p,g)^{14}\mathrm{N}$
r(10) : reaction $^{14}\mathrm{N}(p,g)^{15}\mathrm{O}(e^{+},nu)^{15}\mathrm{N}$
r(11) : reaction $^{15}\mathrm{N}(p,g)^{16}\mathrm{O}$
r(12) : reaction $^{15}\mathrm{N}(p,\alpha)^{12}\mathrm{C}$
r(13) : reaction $^{16}\mathrm{O}(p,g)^{17}\mathrm{F}(e^{+},\nu)^{17}\mathrm{O}$
r(14) : reaction $^{17}\mathrm{O}(p,\alpha)^{14}\mathrm{N}$

References:

Clayton68 : Clayton, Principles of stellar evolution and nucleosynthesis, 1968, p. 380, 392 and 430.

Orignal author(s):

- P.Morel, Département J.D. Cassini, O.C.A.

7.6.2 Routine ppcno10BeBFe

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3α***, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{9}Be, ^{11}B, ^{12}C, ^{13}C, ^{14}N,^{15}N, Ø16,Ø17, ^{56}Fe et l'élément fictif Ex, sont explicitement pris en compte; ^{2}H, ^{7}Be sont supposés à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire la diffusion microscopique des éléments légers et du fer, en tenant compte de l'évolution au long de la pré-séquence principale.

Appel :
ppcno10BeBFe appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

Subroutine ppcno10fe

```
subroutine ppcno10fe( t, ro, comp, dcomp, jac, deriv, fait, epsilon, et, ero, ex, hhe,
↪  be7e, b8e, n13e, o15e, f17e,eps_all_reacs )
```

Private routine of mod_nuc.

PP, CNO cycles with elements : ^{1}H (index 1), ^{3}He (index 2), ^{4}He (index 3), ^{7}Li (index 4), ^{12}C (index 5), ^{13}C (index 6), ^{14}N (index 7), ^{15}N (index 8), ^{16}O (index 9), ^{17}O (index 10), Fe56 (index 11), Ex (index 12). ^{2}H, ^{7}Li, ^{7}Be are considered at equilibrium.

A first call to rq_reac initialise the number of chemical elements for which the nuclear reactions are tabulated in ppcno10. To them, we add Ex, so we have soit nchim+1 elements

Initialise ab_min (negligeable abondances) and ab_ini (initial abondances).

PP chain:
r(1) : reaction $^{1}\mathrm{H}(p,e^{+},\nu)^{2}\mathrm{H}$

r(2) : reaction $^{2}\mathrm{H}(p,g)^{3}\mathrm{He}$
r(3) : reaction $^{3}\mathrm{He}(^{3}\mathrm{He},2p)^{4}\mathrm{He}$
r(4) : reaction $^{3}\mathrm{He}(\alpha,g)^{7}\mathrm{Be}$
r(5) : reaction $^{7}\mathrm{Li}(p,\alpha)^{4}\mathrm{He}$
r(6) : reaction $^{7}\mathrm{Be}(e^{-},\nu g)^{7}\mathrm{Li}$
r(7) : reaction $^{7}\mathrm{Be}(p,g)^{8}\mathrm{B}(e^{+},\nu)^{8}\mathrm{Be}(\alpha)^{4}\mathrm{He}$
CNO cycle:
r(8) : reaction $^{12}\mathrm{C}(p,g)^{13}\mathrm{N}(e^{+},\nu)^{13}\mathrm{C}$
r(9) : reaction $^{13}\mathrm{C}(p,g)^{14}\mathrm{N}$
r(10) : reaction $^{14}\mathrm{N}(p,g)^{15}\mathrm{O}(e^{+},nu)^{15}\mathrm{N}$
r(11) : reaction $^{15}\mathrm{N}(p,g)^{16}\mathrm{O}$
r(12) : reaction $^{15}\mathrm{N}(p,\alpha)^{12}\mathrm{C}$
r(13) : reaction $^{16}\mathrm{O}(p,g)^{17}\mathrm{F}(e^{+},\nu)^{17}\mathrm{O}$
r(14) : reaction $^{17}\mathrm{O}(p,\alpha)^{14}\mathrm{N}$

References:

Clayton68 : Clayton, Principles of stellar evolution and nucleosynthesis, 1968, p. 380, 392 and 430.

Orignal author(s):

- P.Morel, Département J.D. Cassini, O.C.A.

7.6.3 Routine ppcno10K

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3***α, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, Ø16, Ø17 et deux éléments fictifs ExK et Ex, sont explicitement pris en compte; ^{2}H, ^{7}Be sont supposés à l'équilibre, *cf.* §6.10.4 (Page 127). Les deux éléments fictifs représentent respectivement les éléments de masse inférieure et supérieure au potassium. Cette routine a été construite pour décrire la diffusion microscopique différentielle entre les éléments de masse supérieure ou inférieure au potassium.

Appel :
ppcno10K appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.4 Routine ppcno11

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3***α, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{12}C, ^{13}C, ^{14}N, ^{15}N, Ø16, Ø17 et l'élément fictif Ex, sont explicitement pris en compte; ^{7}Be est supposé à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire la pré-séquence principale, en évitant de suivre l'abondance du ^{7}Be dont l'échelle de temps d'évolution est infime.

Appel :
ppcno11 appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.5 Routine ppcno12

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3***α, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{7}Be, ^{12}C, ^{13}C, ^{14}N,^{15}N, Ø16,Ø17 et l'élément fictif Ex, sont explicitement pris en compte; Aucun isotope n'est à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire l'évolution sans élément à l'équilibre.

Appel :

ppcno12 appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.6 Routine ppcno12Be

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3α***, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{7}Li, ^{7}Be, ^{9}Be, ^{12}C, ^{13}C, ^{14}N,^{15}N, Ø16,Ø17 et l'élément fictif Ex, sont explicitement pris en compte; Aucun isotope n'est à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire la diffusion microscopique du beryllium en tenant compte de l'évolution au long de la pré-séquence principale.

Appel :

ppcno12Be appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.7 Routine ppcno12BeBFe

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3α***, *cf.* §7.6.1 (Page 149). Les isotopes ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{7}Be, ^{9}Be, ^{11}B, ^{12}C, ^{13}C, ^{14}N,^{15}N, Ø16,Ø17, ^{56}Fe et l'élément fictif Ex, sont explicitement pris en compte. Aucun isotope n'est à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire la diffusion microscopique des éléments légers, et du fer, en tenant compte de l'évolution au long de la pré-séquence principale.

Appel :

ppcno12BeBFe appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.8 Routine ppcno12Li

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3α***, *cf.* §7.6.1 (Page 149).

Les isotopes ^{1}H, ^{2}H, ^{3}He, ^{4}He, ^{6}Li, ^{7}Li, ^{7}Be, ^{12}C, ^{13}C, ^{14}N, ^{15}N, Ø16,Ø17 et l'élément fictif Ex, sont explicitement pris en compte. Cette routine a été construite pour décrire la diffusion microscopique des éléments légers, plus particulièrement du lithium, en tenant compte de l'évolution au long de la pré-séquence principale. Aucun isotope n'est à l'équilibre, *cf.* §6.10.4 (Page 127).

Appel :

ppcno12Li appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

Subroutine ppcno3a12Ne

```
subroutine ppcno3a12Ne( t, ro, comp, dcomp, jac, deriv, fait,epsilon, et, ero, ex, hhe,
↪  be7e, b8e, n13e, o15e, f17e, eps_all_reacs )
```

Private routine of mod_nuc.

PP, CNO, 3 α and carbon cycles with elements : ^{1}H (index 1), ^{3}He (index 2), ^{4}He (index 3), ^{12}C (index 4), ^{13}C (index 5), ^{14}N (index 6), ^{15}N (index 7), ^{16}O (index 8), ^{17}O (index 9), ^{20}Ne (index 10), Ex (index 11), ^{18}O (index 12). ^{2}H, ^{7}Li, ^{7}Be are considered at equilibrium.

A first call to rq_reac initialise the number of chemical elements for which the nuclear reactions are tabulated in ppcno9a12Ne. To them, we add Ex, so we have soit nchim+1 elements

Initialise ab_min (negligeable abondances) and ab_ini (initial abondances).

PP chain:
r(1) : reaction $^{1}\mathrm{H}(p,e^{+},\nu)^{2}\mathrm{H}$
r(2) : reaction $^{2}\mathrm{H}(p,g)^{3}\mathrm{He}$
r(3) : reaction $^{3}\mathrm{He}(^{3}\mathrm{He},2p)^{4}\mathrm{He}$
r(4) : reaction $^{3}\mathrm{He}(\alpha,g)^{7}\mathrm{Be}$
r(5) : reaction $^{7}\mathrm{Li}(p,\alpha)^{4}\mathrm{He}$
r(6) : reaction $^{7}\mathrm{Be}(e^{-},\nu g)^{7}\mathrm{Li}$
r(7) : reaction $^{7}\mathrm{Be}(p,g)^{8}\mathrm{B}(e^{+},\nu)^{8}\mathrm{Be}(\alpha)^{4}\mathrm{He}$
CNO cycle:
r(8) : reaction $^{12}\mathrm{C}(p,g)^{13}\mathrm{N}(e^{+},\nu)^{13}\mathrm{C}$
r(9) : reaction $^{13}\mathrm{C}(p,g)^{14}\mathrm{N}$
r(10) : reaction $^{14}\mathrm{N}(p,g)^{15}\mathrm{O}(e^{+},nu)^{15}\mathrm{N}$
r(11) : reaction $^{15}\mathrm{N}(p,g)^{16}\mathrm{O}$
r(12) : reaction $^{15}\mathrm{N}(p,\alpha)^{12}\mathrm{C}$
r(13) : reaction $^{16}\mathrm{O}(p,g)^{17}\mathrm{F}(e^{+},\nu)^{17}\mathrm{O}$
r(14) : reaction $^{17}\mathrm{O}(p,\alpha)^{14}\mathrm{N}$
3α **reactions:**
r(15) : reaction $^{4}\mathrm{He}(2\alpha,g)^{12}\mathrm{C}$
r(16) : reaction $^{12}\mathrm{C}(\alpha,g)^{16}\mathrm{O}$
r(17) : reaction $^{16}\mathrm{O}(\alpha,g)^{20}\mathrm{Ne}$
r(18) : reaction $^{14}\mathrm{N}(\alpha,g)F18(e^{+},\nu)^{18}\mathrm{O}$
r(19) : reaction $^{18}\mathrm{O}(\alpha,g)^{22}\mathrm{Ne}$
r(20) : reaction $^{17}\mathrm{O}(p,g)F18(e^{+},\nu)^{18}\mathrm{O}$
r(21) : reaction $^{18}\mathrm{O}(p,\alpha)^{15}\mathrm{N}$
r(22) : reaction $^{20}\mathrm{Ne}(\alpha,g)^{24}\mathrm{Mg}$
Carbon reactions:
r(23) : $^{12}\mathrm{C}(^{12}\mathrm{C},g)^{24}\mathrm{Mg}$
r(24) : $^{12}\mathrm{C}(^{12}\mathrm{C},p)^{23}\mathrm{Na}$
r(25) : $^{12}\mathrm{C}(^{12}\mathrm{C},\alpha)^{20}\mathrm{Ne}$

References:

Clayton68 : Clayton, Principles of stellar evolution and nucleosynthesis, 1968, p. 380, 392 and 430.

Orignal author(s):

- P.Morel, Departement J.D. Cassiopee, O.C.A.

7.6.9 Routine ppcno3a9

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***3***α, *cf.* §7.6.1 (Page 149). Les isotopes ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N,Ø16,Ø17 et l'élément fictif Ex, sont explicitement pris en compte; ^{2}H, ^{7}Li, ^{7}Be sont supposés à l'équilibre, *cf.* §6.10.4 (Page 127).

Appel :
ppcno3a9 appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.10 Routine ppcno3aco

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***co***, *cf.* §7.6.1 (Page 149). Les isotopes ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N,Ø16,Ø17, ^{20}Ne, ^{23}Na, ^{24}Mg, ^{27}Al, ^{28}Si, ¶31,

^{32}S, et l'élément fictif Ex, sont explicitement pris en compte; ^{2}H, ^{7}Li, ^{7}Be sont supposés à l'équilibre, *cf.* §6.10.4 (Page 127). ppcno3aco permet d'atteindre la combustion de l'oxygène.

Appel :

ppcno3aco appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.6.11 Routine ppcno9Fe

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est de type nuc, catégorie ***pré-3***α, *cf.* §7.6.1 (Page 149). Les isotopes ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{13}C, ^{14}N, ^{15}N,Ø16,Ø17, Fe56 et l'élément fictif Ex, sont explicitement pris en compte; ^{2}H, ^{7}Li, ^{7}Be sont supposés à l'équilibre, *cf.* §6.10.4 (Page 127). Cette routine a été construite pour décrire la diffusion microscopique du fer.

Appel :

ppcno9Fe appelée par la routine générique de réactions nucléaires nuc a la même liste d'appel.

7.7 Module mod_cesam

7.7.1 Routine add_ascii

Dans cette routine PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), on complète le tableau des quantités entrant éventuellement dans les fichiers ASCII de type output, *cf.* §7.58 (Page 214).

Description :

En utilisant les valeurs du fichier des oscillations adiabatiques, il y a tabulation, en fonction du rayon, de X pour calcul de la dérivée $\frac{\partial X}{\partial r}$, ainsi que de $F_{\rm rad} = f(r)$ pour celui de $\frac{\partial F_{\rm rad}}{\partial r} = B - J$ et de $\int_0^r \frac{3}{4}\kappa\rho dr = f_{\rm edd}J(r)$. L'ordre des interpolations $m_{\rm x} \geq 2$ fixé dans la routine peut être adapté (ne pas dépasser 4 *i.e.* splines naturelles). Le fichier d'oscillations adiabatiques est ensuite augmenté, soit pour le calcul des inversions (ivar=25) soit pour celui des oscillations non adiabatiques (ivar=44). Les dérivées partielles du second ordre sont déduites numériquement en utilisant la routine df_rotx décrite au §7.15 (Page 170). L'identification des quantités est décrite au §C.1 (Page 345).

Appel : add_ascii est appelée par cesam.

```
1 SUBROUTINE add_ascii(var,glob,itot,ivar)
```

- Entrées :
  - glob: Tableau des quantités globales.
  - itot: Nombre total de points.
  - ivar: Nombre de variables.
- Entrées/Sorties :
  - var: Tableau des variables.

7.7.2 Routine ascii

Cette routine PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), permet la création d'un fichier de sortie ASCII personnalisé. Pour ce faire, coder nom_output='ascii' dans le fichier de données mon_modele.don. Les quantités à écrire, leur ordre et la structure souhaitée sont définis dans un fichier de nom sortie_ascii qui doit se trouver dans l'environnement, *cf.* §?? (Page ??). Les lignes d'identification sont adaptées aux Sorties effectuées. Le fichier ASCII ainsi créé aura l'extension -ascii, Exemple: mon_modele-ascii dont un exemple se trouve dans le sous-directory EXPLOIT.

Diverses possibilités sont offertes :

- Ecriture de la fraction de masse en réel ou en DeX *cf.* §C.1 (Page 345).
- Tabulation du centre vers la surface ou l'inverse.

Les variables et quantités globales à écrire sont **nécessairement** à prendre parmi celles des tableaux `glob` et `var` calculés par `cesam` *cf.* §A.1 (Page 341) et §A.2 (Page 341).

Description :

- Mise en correspondance des tableaux des quantités globales et des tableaux des variables créés par `cesam` et à écrire.
- Transformation éventuelle des masses en DeX ⇒ fraction de masse.
- Inversion éventuelle de l'ordre d'écriture des variables.
- Ecriture des en-têtes.
- Ecriture des variables.

Appel : Le sous-programme `ascii` est appelé par la routine générique `output`, *cf.* §7.58 (Page 214).

```
SUBROUTINE ascii(nglob,tglob,totvar,tvar,var,glob,itot,
1 logm,reverse,titre_ascii)
```

- Entrées :
  - `nglob`: Nombre de quantités globales à écrire.
  - `tglob`: Tableau des indices des quantités globales à écrire.
  - `totvar`: Nombre de variables à écrire.
  - `tvar`: Tableau des indices des variables à écrire.
  - `var`: Tableau des valeurs des variables créé par `cesam`.
  - `glob`: Tableau des valeurs des quantités globales créé par `cesam`.
  - `itot`: Nombre de points du tableau `var`.
  - `logm`=.TRUE.: Tabulation de la masse en DeX, en fraction de masse sinon.
  - `reverse`=.TRUE.: Tabulation du centre vers la surface, de la surface au centre sinon.
  - `titre_ascii`: Nom à attribuer au fichier ainsi créé.

7.7.3 Routine `cesam`

Cette routine PUBLIC, sans argument, du module `mod_cesam`, *cf.* §D.13 (Page 368), constitue en fait le programme principal. Elle gère l'ensemble des calculs; elle est appelée par les programmes `cesam2k` ou `cesam2k_dbg`, *cf.* §7.86 (Page 234). Son organigramme général est présenté Figure **??** (Page **??**). Elle est divisée en 5 parties :

- *Menu d'entrée identifiant l'option de Cesam2k20 à utiliser* :
  - Reprise d'une évolution en cours.
  - Initialisation d'un modèle de séquence principale d'âge zéro homogène.
  - Initialisation d'un modèle de PMS homogène.

  Puis lecture du fichier de données `mon_modele.don`, enfin, initialisation des paramètres de calcul.

- *Cas de la reprise d'une évolution en cours.* Bien qu'il y ait séparation des cas, modèle repris en binaire ou en ASCII, le déroulement des calculs est assez similaire :
  - Lecture du modèle à reprendre, soit en binaire, soit en ASCII.
  - Vérification de la cohérence des paramètres utilisés dans le modèle repris et ceux définis par le fichier de données.
  - Initialisations diverses, en particulier les réactions thermonucléaires et détermination de la valeur du premier pas temporel à utiliser.
  - Pour un modèle repris en ASCII, initialisation des développements sur des bases de B-splines des variables principales et de la composition chimique.
- *Cas de l'initialisation d'un modèle de séquence principale d'âge zéro homogène.*
  - Séparation des cas: modèle d'initialisation en binaire, en ASCII.
  - Pour un modèle repris en ASCII, initialisation des développements sur les bases de B-splines des variables principales.
  - Formation de la fonction de répartition.
  - Initialisations diverses, en particulier les réactions thermonucléaires.
  - calcul du modèle de ZAMS homogène par appel à la routine `resout`.
- *Cas de l'initialisation d'un modèle de PMS homogène.*
  - Séparation des cas: modèle repris en binaire ou en ASCII.
  - Pour un modèle repris en ASCII, initialisation des développements sur des bases de B-splines des variables principales.
  - Formation de la fonction de répartition.
  - Initialisations diverses, en particulier les réactions thermonucléaires.
  - Lecture du facteur de contraction, et calcul d'un premier modèle initial par appel à la routine `resout`.
  - Dialogue de validation du modèle calculé avec, éventuellement, reprise du calcul avec un facteur de contraction amélioré.
  - Calcul d'un second modèle initial avec un facteur de contraction augmenté de 10%.
  - Après validation, calcul du pas temporel initial.
- *Calcul de l'évolution.*
  - Ecriture du modèle en binaire pour les reprises: fichier `mon_modele_B.rep`. En affectant une valeur négative au paramètre `NB_MAX_MODELES` du fichier de données,*cf.* § 3.6 (Page 19), tous les fichiers binaires seront écrits dans l'environnement. Pour identification le numéro du modèle est indiqué dans le nom du fichier, exemple: `mon_modele0146_B.rep`.
  - Mise à jour du fichier `mon_modele.HR` pour dessin du diagramme HR; ce dernier est réinitialisé dans le cas d'une nouvelle évolution.
  - Formation du listing `mon_modele.lis`.
  - Calcul du modèle au pas temporel suivant par appel à la routine `resout`.
  - Gestion du nouveau modèle :
    * Arrêt du calcul, formation du listing, du fichier ASCII s'il est demandé. On trouve au § A.1 (Page 341) et § A.2 (Page 341) la description des tableaux `glob` et `var` créés par `cesam` pour les Sorties ASCII.

* Poursuite du calcul de l'évolution.

Appel :
cesam, appelée par le programme principal, *cf.* §7.86 (Page 234), ***n'a pas d'argument***.

7.8 Module mod_alecian

7.8.1 Routine alecian1

La routine interne alecian1 est une routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), sa fonction est le calcul des accélérations radiatives suivant le premier formalisme de G.Alécian. alecian1 utilise les tables: phi_psi2_100.dat ... phi_psi2_200.dat du sous-directory SUN_STAR_DATA. Chaque fichier correspond à une valeur de la masse clairement identifiée. Cette routine est le regroupement et l'adaptation à Cesam2k20 des données calculées par les sous-programmes modul_ppxi et modul_grad de G.Alécian.

Description :

- Lors du premier appel: lecture des options, des données et préparation de tableaux nécessaires au calcul des accélérations. Les éléments chimiques utilisés sont identifiés et les isotopes sont regroupés. L'accélération radiative de chaque élément non identifié peut, soit être mise à 0, soit être fixée égale à $-g_e$ (gravité), l'élément ne subit alors aucune force externe. Pour chaque élément identifié, bien que le calcul de l'accélération radiative puisse être effectué, l'une ou l'autre des possibilités décrites précédemment peut être éventuellement utilisée. Pour ce faire, il convient de suivre, ***avant compilation***, les intructions données en commentaires. **Problem:** *Dans la présente version de Cesam2k20, les seuls éléments identifiables sont: C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, et Fe. Il n'est pas possible de calculer des accélérations radiatives pour H, He, Li, Be par exemple.*
- Puis, pour chaque appel: calcul des accélérations radiatives et calcul de la gravité effective sur chaque ion.

Appel : alecian1 est appelée par la routine générique f_rad, *cf.* §7.31.21 (Page 195).

```
1 SUBROUTINE alecian1(lum,ray,t,kap,dkapx,nel,ychim,ioni,grav,g_rad,dg_rad)
```

- Entrées :
  - lum, ray, t, kap, dkapx: luminosité, rayon, température, opacité, dérivée/X (mole).
  - nel, ychim, ioni, grav: nombre d'électrons libres par volume, composition chimique par mole, taux d'ionisation, gravité.
- Entrées/Sorties: g_rad(i): vecteur des accélérations radiatives, la gravité est grav+g_rad sur l'élément d'indice i
- Sorties: g_rad(i,j): matrice des dérivées des accélérations radiatives sur l'élément i / abondance par mole de l'élément j.

7.8.2 Routine from_alecian

La subroutine PRIVATE from_alecian du module mod_bp_for_alecian, *cf.* §D.10 (Page 366), regroupe une partie des routines de calcul des accélérations radiatives, selon la seconde version du formalisme développé par G.Alécian. D'origine externe les fonctions des routines de ce regroupement ne sont pas détaillées.

7.9 Module mod_atm

7.9.1 Routine générique atm

Cette routine PUBLIC du module mod_atm, *cf.* §D.8 (Page 364), gère la restitution de l'atmosphère. Elle fait appel à différentes routines PRIVATE, suivant le type d'atmosphère requis défini par la variable nom_atm du fichier de données.

Dans ces routines y sont définies les conditions limites externes du problème différentiel de la structure interne; ces conditions limites doivent porter sur deux des trois variables thermodynamiques P, T et ρ. Elles se trouvent localisées dans la région optiquement mince de l'atmosphère. Les équations de la structure interne supposent que le flux d'énergie radiative est transporté par diffusivité radiative *i.e.* approximation de diffusion, ce qui est inexact dans le milieu optiquement mince où sont définies les conditions limites externes. Il faut donc transporter ces limites du milieu mince dans le milieu épais. La source de Cesam2k20 offre la possibilité d'utiliser deux types de méthodes: l'approximation monocouche et la reconstitution de l'atmosphère. Voir, pour plus de détails, Morel et al. (1994).

Appel : atm est appelée par les routines cesam, static_m et static_r.

```
1 SUBROUTINE atm(list,l_rac,r_rac,xchim,pt_rac,dptsdl,dptsdr,
2     1 t_rac,dtsdl,dtsdr,m_rac,dmsdl,dmsdr,p_rac,dpsdl,dpsdr,t_eff)
```

- Entrées :
  - list=.TRUE.: calcul réduit pour une liste (sans les dérivées),
  - l_rac: luminosité au raccord,
  - r_rac: rayon au raccord,
  - xchim : composition chimique par gramme.
- Sorties :
  - pt_rac: pression totale au raccord,
  - dptsdl: dérivée / L de la pression totale au raccord,
  - dptsdr: dérivée / R de la pression totale au raccord,
  - t_rac: température au raccord,
  - dtsdl: dérivée / L de la température au raccord,
  - dtsdr: dérivée / R de la température au raccord,
  - m_rac: masse au raccord,
  - dmsdl: dérivée / L de la masse au raccord,
  - dmsdr: dérivée / R de la masse au raccord,
  - p_rac: pression gazeuse au raccord,
  - dpsdl: dérivée / L de la pression gazeuse au raccord,
  - dpsdr: dérivée / R de la pression gazeuse au raccord,
  - t_eff : température effective.

7.9.2 Routine `coll_atm`

La fonction de cette routine PRIVATE du module `mod_atm`, *cf.* §D.8 (Page 364), est la résolution, dans l'espace des B-splines, du système des équations de la restitution de l'atmosphère Eq. 6.57 (Page 71).

Description :

- Initialisation du vecteur nodal et des limites, initialisations et allocations diverses lors du premier appel.
- Vérification, qu'au point de raccord, l'accélération centrifuge ne dépasse pas la gravité.
- Pour chaque point de collocation et chaque point limite, formation par appels à `eq_atm` des Eq. 6.57 (Page 71) dans l'espace des B-splines.
- Résolution du système et évaluation des corrections de la méthode itérative, retour à `lim_atm` ou poursuite des itérations (boucle infinie).

Appel : `coll_atm` est appelée par la routine `lim_atm` *cf.* §7.10.2 (Page 166).

```
SUBROUTINE coll_atm(r_rac,l_rac,xchim,ord_atm,knot_atm,dim_atm,ms_atm)
```

- Entrées :
  - `r_rac, l_rac, xchim`: rayon, luminosité composition chimique au raccord avec l'enveloppe.
  - `ord_atm, knot_atm, dim_atm, ms_atm`: ordre des équations différentielles, nombre de points du vecteur nodal, dimension de la base de B-splines, ordre des B-splines.

7.10 Routine générique `tdetau`

Cette routine PUBLIC du module `mod_atm`, *cf.* §D.8 (Page 364), constitue la routine générique de la gestion du calcul de la température en fonction de la profondeur optique.

Description :

Le calcul est orienté vers la routine de loi $T(\tau)$ indiquée dans le fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14).

Appel :

Les lois $T(\tau)$ étant utilisées pour reconstruire une atmosphère, la routine `tdetau` est appelée de `lim_atm`, *cf.* §7.10.2 (Page 166).

```
SUBROUTINE tdetau(tau,teff,grav,t,dtsdtau,dtsdteff,dtsdg,
    1 ro_ext,dro_grav,dro_teff,f_tau,df_tau,d2f_tau)
```

- Entrées :
  - `tau, teff, grav`: profondeur optique, température effective, gravité.
- Sorties :
  - `t, dtsdtau, dtsdteff, dtsdg`: température et dérivées;
  - `ro_ext, dro_grav, dro_teff`: densité externe et dérivées;
  - `f_tau, df_tau, d2f_tau`: fonction f et dérivées *cf.* §6.3.3 (Page 69).

Routine edding

Cette routine PRIVATE du module mod_atm, *cf.* §D.8 (Page 364), calcule la température dans une atmosphère purement radiative, pour une profondeur optique grise donnée suivant la loi $T(\tau)$ d'Eddington.

Appel :
edding, appelée par la routine générique tdetau *cf.* §7.10 (Page 162), utilise les mêmes arguments.

Subroutine roger

```
subroutine roger( tau, teff, grav, t, dtsdtau, dtsdteff, dtsdg, ro_ext, dro_grav,
↪  dro_teff, f_tau, df_tau, d2f_tau)
```

Private routine of module mod_atm.
$T(\tau]$ relations derived by R. Cayrel from the Kurucz's ATLAS12 program.

Arguments:

tau : **real**(kind=dp), **intent**(in)
Rossland oprical depth.

teff : **real**(kind=dp), **intent**(in)
Effective temperature.

grav : **real**(kind=dp), **intent**(in)
GRavity.

t : **real**(kind=dp), **intent**(out)
Temperature.

dtsdtau : **real**(kind=dp), **intent**(out)
Derivative of temperature with respect to τ.

dtsdteff : **real**(kind=dp), **intent**(out)
Derivative of temperature with respect to $T_{\rm eff}$.

dtsdg : **real**(kind=dp), **intent**(out)
Derivative of temperature with respect to gravity.

ro_ext : **real**(kind=dp), **intent**(out)
External density.

dro_grav : **real**(kind=dp), **intent**(out)
Derivative of ρ with respect to gravity.

dro_teff : **real**(kind=dp), **intent**(out)
Derivative of ρ with respect to $T_{\rm eff}$.

f_tau, df_tau, d2f_tau : **real**(kind=dp), **intent**(out)
Function f and derivatives wrt τ. Cf. Section on Numerical techniques used for the connection of ∇ in the documentation.

Orignal author(s):

- P. Morel, Département J.D. Cassini, O.C.A.
- B. Pichon, CESAM2k.

Routines k5750, k5777

Ces deux routines PRIVATE du module mod_atm, *cf.* §D.8 (Page 364), calculent la température dans une atmosphère non purement radiative pour une profondeur optique grise donnée, suivant les lois $T(\tau)$ solaires $T_{\rm eff} = 5750\,\mathrm{K}$ et $T_{\rm eff} = 5777\,\mathrm{K}$ calculées par C. Van't Veer avec le programme Atlas 9 de Kurucz.

Description :

- Mise en forme des données pour l'interpolation.
- Interpolation de la température et dérivées.

Appel :
k5750, k5777, sont appelées par la routine générique tdetau, *cf.* §7.10 (Page 162). Elles utilisent les mêmes arguments.

Subroutine ball21

```
subroutine ball21( tau, teff, grav, t, dtsdtau, dtsdteff, dtsdg, ro_ext, dro_grav,
↪  dro_teff, q_tau, dq_tau, d2q_tau )
```

Private routine of mod_atm module. $T(\tau)$ relation derived by Ball (2021) by fitting data fron Trampedach et al. 2014

Arguments:

tau : real(kind=dp), intent(in)
Rosseland optical depth.

teff : real(kind=dp), intent(in)
Effective temperature.

grav : real(kind=dp), intent(in)
Gravity.

t : real(kind=dp), intent(out)
Temperature.

dtsdtau : real(kind=dp), intent(out)
Tau derivative of temperature.

dtsdteff : real(kind=dp), intent(out)
Teff derivative of temperature.

dtsdg : real(kind=dp), intent(out)
Gravity derivative of temperature.

ro_ext : real(kind=dp), intent(out)
External density.

dro_grav : real(kind=dp), intent(out)
Gravity derivative of density.

dro_teff : real(kind=dp), intent(out)
Teff derivative of density.

q_tau : real(kind=dp), intent(out)
Q(tau).

dq_tau : **real**(kind=dp), **intent**(out)
$\mathrm{d}q(\tau)/\mathrm{d}\tau$.

d2q_tau : **real**(kind=dp), **intent**(out)
$\mathrm{d}^2q(\tau)/\mathrm{d}\tau^2$.

History:

2021-11 : Initial implementation (Louis Manchon).

Routine hopf

Cette routine PRIVATE du module mod_atm, *cf.* §D.8 (Page 364), calcule la température dans une atmosphère purement radiative pour une profondeur optique grise donnée suivant la loi $T(\tau)$ de Hopf, *cf.* **?**.

Description :

- Mise en forme des données pour l'interpolation.
- Interpolation de la température et dérivées.

Appel :
hopf, appelée par la routine générique tdetau *cf.* §7.10 (Page 162), utilise les mêmes arguments.

7.10.1 Routine eq_atm

La fonction de cette routine PRIVATE du module mod_atm, *cf.* §D.8 (Page 364), est de former les équations de restitution de l'atmosphère Eq. 6.57 (Page 71).

Description :

- Au premier appel, après quelques initialisations, on projette sur la base de B-spline l'atmosphère solaire, connue sous forme de DATA, on obtient ainsi la solution initiale, on définit ensuite les limites et on détermine les points de collocation.
- Extraction des grandeurs physiques, calcul de la T_{eff}, de la gravité et loi $T(\tau)$.
- Calcul des grandeurs thermodynamiques, de ∇ en particulier, et dérivées, formation des équations aux points de collocation et aux points limites.

Appel :
eq_atm est appelée de coll_atm, *cf.* §7.9.2 (Page 162) :

```
1 SUBROUTINE eq_atm(fait,li,xchim,cx,y,be,ae,r_rac,l_rac,xcoll_atm)
```

- Entrées :
  - fait, li, xchim, cx: fait=1 point de collocation, fait=2 point limite, indice de la limite, composition chimique/gramme, indice du point de collocation,
  - y: variables et dérivées,
  - r_rac,l_rac, xcoll_atm: rayon au fond de l'atmosphère, luminosité, abscisses des points de collocation.
- Sorties :
  - be, ae: seconds membres et jacobien.

7.10.2 Routine `lim_atm`

Cette routine PUBLIC du module `mod_atm`, *cf.* §D.8 (Page 364), permet de restituer l'atmosphère à partir d'une loi $T(\tau)$ grise, *cf.* §6.3.2 (Page 67).

Description :

- Au premier appel :
  - Initialisation de constantes.
  - Recherche, dans l'environnement, d'un fichier binaire d'atmosphère pour reprise, et tests de cohérence.
  - Reprise de l'atmosphère en binaire ou initialisation en ASCII.
  - Recherche du rayon bolométrique et initialisations diverses.
  - Projection sur la base de B-splines.
- Pour chaque appel :
  - Commentaires divers.
  - Appel à `coll_atm` pour l'intégration.
  - Au cas où une liste complète du modèle est requise, formation des tableaux des quantités nécessaires.
  - Ecriture du fichier binaire d'atmosphère `mon_modele.atm`.

Appel :

`lim_atm` appelée par la routine générique d'atmosphère `atm`, *cf.* §7.9.1 (Page 161), a la même liste d'appel.

7.10.3 Routines `lim_gong1, lim_tau1`

Ces deux routines PUBLIC du module `mod_atm`, *cf.* §D.8 (Page 364), sont deux formes similaires de la restitution d'une atmosphère dans l'approximation monocouche, *cf.* §6.3.1 (Page 66). Dans la routine `lim_gong1`, deux facteurs $\beta = 7.22$ et $\lambda = 6$, affectent respectivement les équations de la pression et de la luminosité, leur effet étant, avec une Equation of State supposant l'ionisation totale, d'obtenir un modèle solaire qui ressemble au soleil actuel.

Problem: *Avec l'approximation monocouche, la pression turbulente est ignorée dans l'atmosphère.*

Appel :

`lim_gong1, lim_tau1`, appelées par la fonction générique d'atmosphère `atm`, *cf.* §7.9.1 (Page 161), ont la même liste d'appel.

7.11 Module `mod_variables`

7.11.1 Routine `chim_gram`

La fonction de cette routine PUBLIC du module `mod_variables` effectue la transformation des valeurs des rapports d'abondances par mole et de leurs dérivées en rapports d'abondances par masse.

Appel : `chim_gram` est appelée de divers endroits, depuis `cesam` particulièrement.

```
1   SUBROUTINE chim_gram(xchim,dxchim)
```

- Entrées/Sorties :
  - `xchim, dxchim`: composition chimique et dérivée par mole $\longrightarrow$ par gramme.

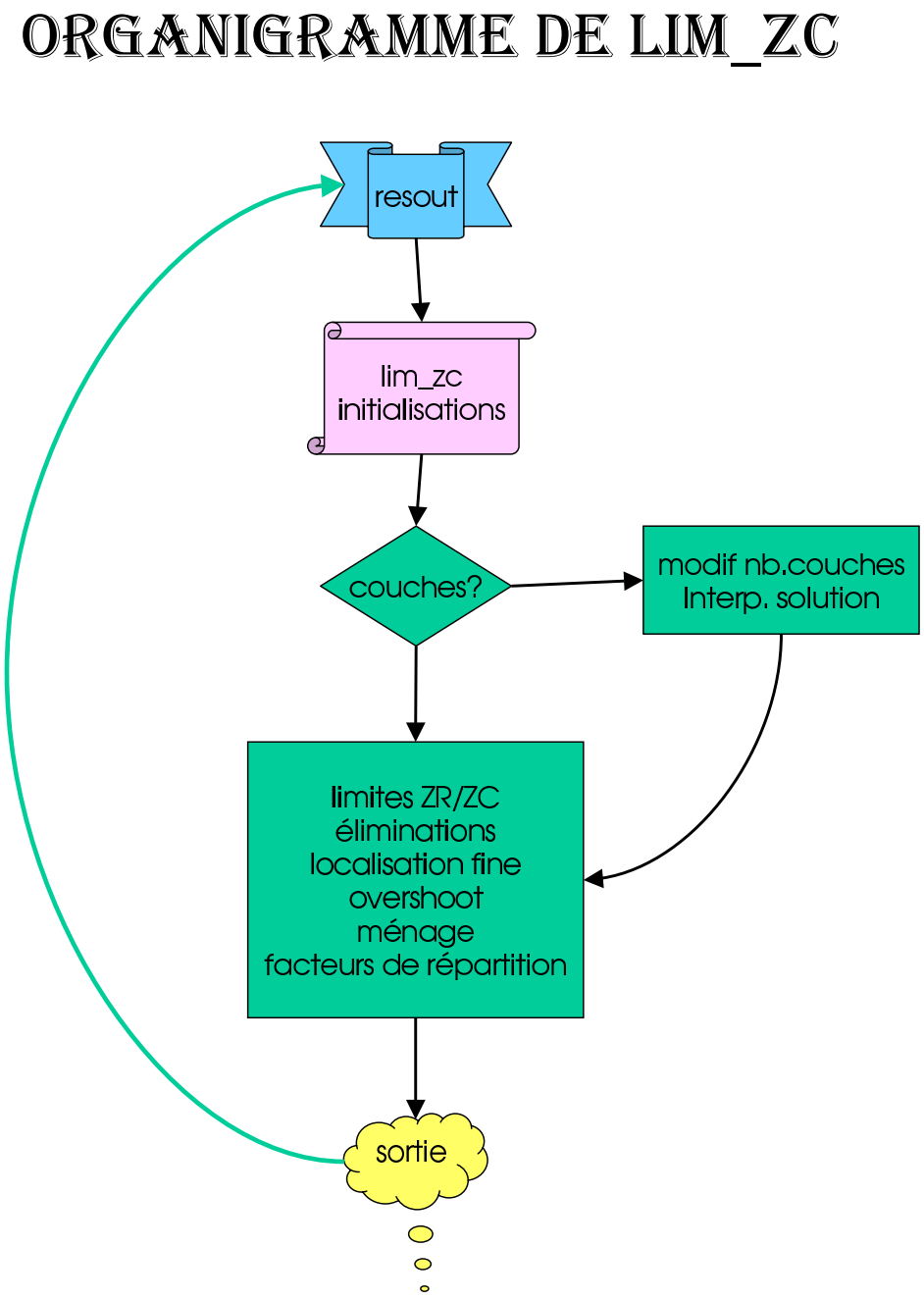


Figure 7.1: Organigramme de la routine `lim_zc` de détermination des limites zones radiatives / zones convectives.

7.12 Module mod_conv

7.12.1 Subroutine alpha_guess

```
subroutine alpha_guess( )
```

Performs bilinear interpolation of alpha (1st guess) with given values at two different values of mass and mean mol. Weight.

Routine of mod_conv

History:

2022 : Initial implementation (Louis Manchon).

2023-4 : Implementation of the bilinear interpolation (Louis Manchon).

7.12.2 Routine générique conv

Cette routine PUBLIC du module mod_conv, *cf.* §D.7 (Page 364), gére le calcul du gradient de température $\nabla \equiv \frac{\partial \ln T}{\partial \ln P}$ dans les zones convectives et de ses dérivées partielles par rapport aux variables locales. Suivant la formulation de la convection définie dans le fichier de données, *cf.* §3.3 (Page 14), un appel ciblé dirige le calcul vers la routine *ad-hoc*.

Description :

La structure des routines de convection est la même pour toutes. Seules diffèrent les formulations *cf.* §6.9.2 (Page 118). Des grandeurs spécifiques aux conditions physiques du calcul sont initialisées au premier appel. Puis la racine réelle de la cubique est recherchée en utilisant l'algorithme de Newton-Raphson. Enfin le gradient convectif, l'efficacité de la convection et leurs diverses dérivées sont calculés.

Appel : Le sous-programme conv est appelé par les routines thermo, *cf.* §7.80 (Page 229), et thermo_atm, *cf.* §7.81 (Page 230).

```
1 SUBROUTINE conv(r,krad,gravite,delta,cp,ro,hp,taur,gradrad,
2     1 gradad,der,gradconv,dgradkra,dgradgra,dgradel,dgradcp,dgradro,
3     2 dgradhp,dgradtaur,dgradgrad,dgradgad,
4     3 gam,dgamkra,dgamgra,dgamdel,dgamcp,dgamro,
5     4 dgamhp,dgamtaur,dgamgrad,dgamgad)
```

- Entrées :
  - r, krad, gravite, delta, cp: rayon, conductivité radiative, gravité, δ, $c_{\rm p}$.
  - ro ,hp, taur, gradrad, gradad: densité, échelle de hauteur de pression, épaisseur optique de la bulle convective, gradient radiatif, gradient adiabatique - éventuellement corrigé du terme $\frac{\mathrm{d}\ln P_{\rm gaz}}{\mathrm{d}\ln P}$, *cf.* §6.9.3 (Page 119).
  - der=.TRUE.: le calcul des dérivées est requis.
- Sorties :
  - gradconv, dgradkra, dgradgra, dgradel, dgradcp, dgradro, dgradhp, dgradtaur, dgradrad, dgradad: gradient de température et dérivées.
  - gam, dgamkra, dgamgra, dgamdel, dgamcp, dgamro, dgamhp, dgamtaur, dgamgrad, dgamgad: efficacité de la convection et dérivées.

Routine conv_a0

En un point d'une zone convective, il semble irréaliste que la longueur de mélange puisse être supérieure à la distance qui sépare ce point de la plus proche limite de la zone convective. Ainsi que **?** l'a proposé, avec la routine PRIVATE conv_a0 du module mod_conv, *cf.* §D.7 (Page 364), la longueur de mélange utilisée devient nulle à chaque limite RZ/CZ. L'introduction de la distance exacte change la nature du problème qui devient intégro-différentiel ce qui complique considérablement la résolution numérique. Ce qu'évite la formulation locale d'Eggleton, *cf.* §6.9.2 (Page 118).

Routine conv_cm

Dans la routine PRIVATE conv_cm du module mod_conv, *cf.* §D.7 (Page 364), le gradient de température dans une zone convective est calculé selon les prescriptions de Canuto & Mazzitelli (1991) avec, pour longueur de mélange, $l = \alpha H_{\rm p}$, *cf.* §6.9.2 (Page 118).

Routines conv_cgm_reza, conv_cm_reza

Dans la routine PRIVATE conv_cm_reza du module mod_conv, *cf.* §D.7 (Page 364), similaire à conv_cm, il est tenu compte de la quantité thermodynamique $\delta = (\frac{\partial \ln \rho}{\partial \ln T})_P$. Dans la routine conv_cgm_reza, le calcul du gradient convectif est effectué selon les prescriptions de Canuto et al. (1996) avec la prescription de Bernkopf (**?**), *cf.* §6.9.2 (Page 118).

Ces deux routines ont été mises en oeuvre et mises à la disposition des utilisateurs de Cesam2k20 par Reza Samadi, LESIA , Observatoire de Paris

Routine conv_jmj

Les spécificités de la routine de convection PRIVATE conv_jmj du module mod_conv, *cf.* §D.7 (Page 364), ont été décrites au §6.9.2 (Page 118). La mise en oeuvre de conv_jmj a bénéficié de la collaboration de J.Provost et de M.J.Goupil.

7.13 Routine générique des

Cette routine PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), est la ***routine générique*** de dessin, elle utilise le logiciel pgplot *cf.* §**??** (Page **??**). Au fur et à mesure du calcul, le diagramme HR, la répartition en RZ/CZ, les variables de structure et les abondances en fonction de la masse ou du rayon sont tracés.

Appel :
des qui n'a que des arguments d'entrée est appelée par cesam.

```
1 SUBROUTINE des(fin,dt,teff)
```

- fin=.TRUE.: dernier appel, fermeture du dessin,
- dt, teff: pas temporel, température effective.

7.14 Routines des_m, des_r

L'une ou l'autre de ces routines PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), est appelée par la routine générique des suivant le type de dessin défini par la variable nom_des du fichier de données, *cf.* §3.3 (Page 14). Le dessin est effectué respectivement en fonction de la masse (nom_des='des_m') ou du rayon nom_des='des_r'), ou encore n'est pas effectué (nom_des='no_des')

Description :
Les algorithmes de ces deux routines sont similaires :

- Initialisations diverses relatives au logiciel PGPLOT.
- Tracé des abondances par unité de masse, ^{1}H et ^{4}He sont dessinées en valeurs réelles, les abondances des autres éléments sont normalisées sur [0,1], les coefficients de normalisation sont indiqués sur l'écran pour chaque variable.
- Tracé des variables de structure normalisées, pression, température, rayon ou masse et luminosité, les coefficients de normalisation sont indiqués sur l'écran pour chaque variable.
- Tracé du diagramme HR, sur les deux axes, les échelles sont déterminées lors du premier appel en fonction de la valeur de la masse initiale; si le fichier mon_modele.TeffL existe dans l'environnement, tracé de la boîte d'erreur *cf.* §?? (Page ??).
- Tracé des zones convectives, à l'issue de chaque pas temporel, un trait plein indique les intervalles de masse affectés par la convection.

Particularité de des_r: Le rayon d'une étoile change au cours de l'évolution; il arrive que les graphes de la représentation des variables de structure en fonction du rayon utilisent mal le domaine spatial de l'épure. Lorsque le rayon total devient, soit supérieur aux 9/10, soit inférieur aux 3/4, du domaine, on effectue un changement d'échelle qui a pour effet de ramener la surface aux 3/4 du domaine. Tant que l'échelle en rayon reste fixée, un tiret vertical - aux 3/4 du domaine - matérialise la valeur antérieure du rayon qui a servi à définir l'échelle. Cette disposition permet de suivre l'évolution du rayon en restant dans le cadre de l'épure.

7.15 Routine df_rotx

Cette routine PUBLIC du module mod_etat, *cf.* §D.5 (Page 363), constitue un opérateur qui permet de transformer les dérivées par rapport à P, T, X d'une fonction thermodynamique $f(P, T, X)$ en celles par rapport à ρ, T, X. En appliquant plusieurs fois l'opérateur on obtient les dérivées d'ordre supérieur.

Appel :
df_rotx est appelée par add_ascii, *cf.* §7.7.1 (Page 157).

```
1 SUBROUTINE df_rotx(p,ro,t,alfa,delta,dfdp_tx,dfdt_px,dfdx_pt,
2   1 drodx_pt,df_dro,df_dt,df_dx)
```

- Entrées :
  - p, ro, t, alfa, delta: pression, densité, température, $\frac{\partial \ln \rho}{\partial \ln P}$, $-\frac{\partial \ln \rho}{\partial \ln T}$,
  - dfdp_tx, dfdt_px, dfdx_pt,drodx_pt: $\frac{\partial f}{\partial P}$, $\frac{\partial f}{\partial T}$, $\frac{\partial f}{\partial X}$, $\frac{\partial \rho}{\partial X}$.
- Sorties :
  - df_dro, df_dt, df_dx: $\frac{\partial f}{\partial \rho}$, $\frac{\partial f}{\partial T}$, $\frac{\partial f}{\partial X}$

7.16 Fonction dgrad

Cette fonction PUBLIC du module mod_static, *cf.* §D.12 (Page 367), forme le critère de convection de Schwarzschild ou de Ledoux suivant les prescriptions décrites au §6.9.1 (Page 117).

Description :

- Après quelques initialisations, appel à l'Equation of State et calcul de l'opacité
- Calcul du gradient radiatif avec le cas particulier du centre.
- Formation du critère de convection de Schwarzschild ou de Ledoux.

Problem: *Pour une raison inconnue, l'utilisation du critère de Ledoux est délicate.*

Appel :
dgrad est appelée de lim_zc, *cf.* §7.42 (Page 204).

```
1   FUNCTION dgrad(pt,p,t,dlpp,xchim,m,l,r,dxchim,w)
```

- pt, p, t, dlpp, xchim, dxchim: Pression totale et gazeuse, température, $\frac{\partial \ln P_{\rm gaz}}{\partial \ln P}$, composition chimique et dérivée;
- m, l, r, w: masse, luminosité, rayon, vitesse angulaire.

7.17 Routine diffus

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), gère l'intégration numérique des équations de diffusion des espèces chimiques *cf.* §6.7 (Page 84) et du moment cinétique *cf.* §6.8 (Page 98).
Ces équations, du type intégro-différentiel, sont couplées. Pour des raisons d'instabilité numérique, il s'est avéré extrèmement délicat de les résoudre simultatément, force a été de les traiter séquentiellement, au prix d'une convervenge médiocre. Leur résolution numérique utilise la méthode des éléments finis de Galerkin, *cf.* §**??** (Page **??**). Bien que la structure des algorithmes de résolution utilisés pour chacun des deux systèmes différentiels soient identiques deux routines distinctes ont été créées, pour conserver un maximum de souplesse.

Description :

- Au premier appel, initialisation et diverses écritures.
- Recherche du nombre et des indices des limites RZ/CZ.
- Création du vecteur nodal pour la diffusion des éléments chimiques.
- Création du vecteur nodal pour la diffusion du moment cinétique.
- Résolution des équations de la diffusion du moment cinétique.
- Résolution des équations de la diffusion des éléments chimiques.
- Déallocations diverses.

Appel :
diffus est appelée par evol, *cf.* §7.31.3 (Page 183).

```
SUBROUTINE diffus(ok,dt,mc_tmp,nc_tmp)
```

- Entrées :
  - `dt`: pas temporel,
  - `mc_tmp,nc_tmp`: masses temporaires (abscisses langrangiennes), nombre de masses temporaires.
- Sorties :
  - `ok`: `ok=.TRUE.` les algorithmes de résolution de la diffusion du moment cinétique et des espèces chimiques ont convergé au niveau de précision requis.

7.18 Routine `dnunl`

La fonction de cette routine PRIVATE du module `mod_cesam`, *cf.* §D.13 (Page 368), est le calcul approximatif, valable uniquement pour le cas solaire, de ν_0, $\delta\nu_{02}$ et $\delta\nu_{13}$ suivant Schatzman & Praderie **?**, eq. 101-102 .

Description :

- Tabulation de la vitesse du son en fonction de R^2, calcul de ν_0 par intégration numérique de type Gauss (on utilise le fait que $rdr = 1/2dr^2$).
- Calcul de $\delta\nu_{02}$ et $\delta\nu_{13}$.

Appel :
`dnunl` est appelée de `list`, *cf.* §7.43 (Page 205).

```
SUBROUTINE dnunl(r,c,n,rtot,nu0,dnu02,dnu13,a)
```

- Entrées :
  - `r, c, n, rtot`: tables des rayons, des vitesses du son, nombre de points, rayon total.
- Sorties :
  - `nu0, dnu02, dnu13, a`: ν_0, $\delta\nu_{02}$, $\delta\nu_{13}$, le A de la formule 100.

7.19 Routine subordonnée `ecrit_ascii`

Cette routine subordonnée de la routine `cesam` a pour fonctions la formation la mise en forme et l'écriture des fichiers de sortie ASCII, *cf.* §3.4 (Page 17). Pour décrire finement le comportement de la fréquence de **v**, dans les fichiers de sortie destinés aux exploitations sismologiques, *cf.* §6.9.9 (Page 121), des points sont ajoutés au voisinage des limites zones radiatives / zones convectives; typiquement, avec la valeur du paramètre $l_0 = 3$, 20 points supplémentaires sont ajoutés de part et d'autre de chaque limite.

Description :

- Au premier appel, allocation du tableau des quantités globales.

- Allocations de tableaux et création des abscisses supplémentaires permettant de décrire finement le profil de la fréquence de **v** au voisinage des limites zones radiatives / zones convectives. Par défaut, on dispose 10 abscisses supplémentaires de part et d'autre de chaque limite. Insertion de ces abscisses dans le tableau des abscisses connues.
- Formation de la table des quantités globales.
- Allocation du tableau des quantités variables et formation de la partie atmosphère.
- Formation de la partie structure interne et composition chimique. Les quantités correspondants aux abscisses déjà connues sont reprises; celles correspondant aux point ajoutés sont calculées.
- Déallocations de tableaux.
- Appel à la routine `add_ascii`, *cf.* §7.7.1 (Page 157) pour complément de données.
- Appel à la routine générique `output`, *cf.* §7.58 (Page 214), qui oriente vers le type de sortie ASCII désirée.
- Déallocation du tableau des variables.

Appel :
`écrit_ascii`, qui n'a pas d'argument, est appelée de `cesam` à l'occasion de la sortie de fin de calcul, ou encore à l'issue de chaque pas temporel si on désire conserver tous les fichiers ASCII, *cf.* §3.4 (Page 17).

7.20 Routine subordonnée `ecrit_rota`

Cette routine subordonnée de `diffus` et de `evol`, a pour fonction l'écriture, dans l'évironnement du calcul, du fichier des variables et des coefficients des équations de la diffusion du moment cinétique, *cf.* §6.8 (Page 98). Ce fichier est destiné à être exploité par le programme de dessin `des_rota.f90` du sous-directory `EXPLOIT`, *cf.* §**??** (Page **??**). Ce fichier de travail n'est pas destiné à des exploitations mais aux mises au point. Il n'est créé que si la variable logique `ecrit_rot`, *cf.* §3.16 (Page 33), du module `mod_donnees` est vraie, il est alors formé à l'issue de chaque appel à la routine `resout_rota`, *cf.* §7.69 (Page 221).

Description :

- Définition du nombre de variables et allocation du tableau à écrire.
- Calcul des coefficients des équations de la diffusion du moment cinétique, aux abscisses du vecteur de la rotation (le centre est omis en raison de la singularité de l'échelle de hauteur de presion).
- Formation du tableau des variables et écriture sur le fichier `mon_modele_coeff_rota.dat`. Ce dernier est écrasé par chaque nouvelle écriture.

Appel : Le sous-programme `ecrit_rota`, n'a pas d'argument.

7.21 Routine `eq_diff_chim`

La routine PRIVATE `eq_diff_chim` du module `mod_evol`, *cf.* §D.11 (Page 367), a pour fonction de calculer, dans l'espace physique, les résidus des équations 6.128 (Page 86), les quantités figurant à gauche des produits scalaires et leurs dérivées par rapport aux abondances.

Pour un élément chimique d'indice i fixé, les coefficients de l'équation de diffusion calculés dans les routines de type `diffm`, *cf.* §7.31.14 (Page 189) sont de trois sortes:

- Les coefficients v_i de x_i, dans l'équation 6.117 (Page 84), codés `v(i)`.
- Les coefficients de diffusion microscopique $d^{\star}_{i,j}$, de $\frac{\partial x_j}{\partial m}$, dans l'équation 6.119 (Page 84), codés `d(i,j)`.
- Les coefficients $d_{i,i}$, de $\frac{\partial x_j}{\partial m}$, dans l'équation 6.119 (Page 84), sont ajoutés à `d(i,i)` dans les routines de type `difft`, *cf.* §7.31.8 (Page 186).

les dérivées par rapport à $x_k,\ k = 1, n_{\rm elem}$ sont respectivement notées `dv(i,k) = dv(nbelem(k-1)+i)` et `dd(i,j,k) = dd(nbelem(nbelem(k-1)+j-1)+i)`.

Description :

- Au point courant :
  - Initialisations diverses,
  - extraction des variables de structure au point de calcul `cx`, appel à l'Equation of State, détermination des opacités, calcul des réactions thermonucléaires et des coefficients de l'équation de diffusion,
  - détermination des résidus des équations 6.128 (Page 86), des quantités figurant à gauche des produits scalaires et de leurs dérivées par rapport aux abondances.
- Au point limite externe, évaluation et exploitation de la perte de moment angulaire.

Appel :
`eq_dif_chim` est appelée par `resout_chim`, *cf.* §7.68 (Page 220).

```
1 SUBROUTINE eq_diff_chim(nu,y,dt,ad,as,bd,bs)
```

- Entrées :
  - `nu, dt, y`: point de calcul en $m^{\frac{2}{3}}$, pas temporel, variables et dérivées.
- Sorties :
  - `as, ad, bs, bd`: dérivées par rapport aux abondances $x_{i,j}$ des arguments des produits scalaires avec $S_j^{m_c}$ et $\frac{\partial x_{i,j}}{\partial \nu}$ et résidus.

7.22 Routine `eq_diff_poisson`

La routine PRIVATE `eq_diff_poisson` du module `mod_evol`, *cf.* §D.11 (Page 367), a pour fonction de calculer, dans l'espace physique, les coefficients des équations, à résoudre dans `poisson_initial.f90`, *cf.* §7.64 (Page 216), pour l'initialisation du potentiel gravitationnel.

Description :

- initialisation de constantes et allocations.
- calcul des coefficients au temps $t + dt$.
- formation des coefficients au point courant.
- formation des coefficients aux points limites.

Appel :
`eq_diff_poisson` est appelée par `poisson_initial`, *cf.* §7.64 (Page 216).

```
SUBROUTINE eq_diff_poisson(fait,nu,ad,as,bs)
```

- Entrées :
  - fait, nu: fait=0: point courant, fait=1, 2: limite, point de calcul en $m^{\frac{2}{3}}$, variables et dérivées.
- Sorties :
  - as, ad, bs: dérivées par rapport à la variable Ψ des arguments des produits scalaires avec $S_j^{m_c}$ et $\frac{\partial x_{i,j}}{\partial \nu}$, résidus.

7.23 Routine eq_diff_rota3/4

La routine PRIVATE eq_diff_rota du module mod_evol, *cf.* §D.11 (Page 367), a pour fonction de calculer, dans l'espace physique, les coefficients et les résidus des équations de la diffusion du moment cinétique, §6.8 (Page 98).

Description :

- Interpolation au point d'abscisse lagrangienne nu des coefficients des équations.
- Détermination de la nature du milieu convectif ou radiatif par interpolation du vecteur de convection.
- Au point courant (fait=1) :
  - Formation des équations dans le milieu convectif.
  - Formation des équations dans le milieu radiatif.
- Sur les limites (fait=2, 3) :
  - Partie intégrée des équations pour un noyau convectif et une zone convective externe.

Appel :
eq_rota est appelée par resout_rota, *cf.* §7.69 (Page 221).

```
SUBROUTINE eq_diff_rota(fait,nu,y,ad,as,bd,bs)
```

- Entrées :
  - fait, nu, y: fait=1: point courant, fait=2, 3: limite, point de calcul en $m^{\frac{2}{3}}$, variables et dérivées.
- Sorties :
  - as, ad, bs, bd: dérivées par rapport aux variables Ω, U, θ, Λ, ψ des arguments des produits scalaires avec $S_j^{m_c}$ et $\frac{\partial x_{i,j}}{\partial \nu}$, résidus.

7.24 Module mod_etat

7.24.1 Routine générique etat

Les routines de type etat constituent les interfaces entre Cesam2k20 et la routine de calcul de l'Equation of State qui a souvent une origine externe. Ces routines sont des routines PRIVATE du module mod_etat, *cf.* §D.5 (Page 363).

Pour une pression gazeuse P, une température T et une composition chimique X_i données, etat calcule la densité ρ, l'énergie interne spécifique U, $c_{\rm p}$, δ, $\nabla_{\rm ad}$, et leurs dérivées premières, α, β, Γ_1. Certaines routines d'équation d'état utilisent des tables que l'on doit créer dans le sous-directory SUN_STAR_DATA, *cf.* §?? (Page ??). Le nom de ces tables doit être indiqué dans le fichier de données. La ***routine générique*** de calcul de l'équation d'état est etat, routine PUBLIC du module mod_etat.

Appel :

etat est appelée de divers endroits, en particulier depuis evol, thermo, thermo_atm, eq_dif_chim, cesam.

```
SUBROUTINE etat(p,t,xchim,deriv,
     1 ro,drop,drot,drox,u,dup,dut,dux,
     2 delta,deltap,deltat,deltax,cp,dcpp,dcpt,dcpx,
     3 gradad,dgradadp,dgradadt,dgradadx,alfa,beta,gamma1)
```

- Entrées :
  - p, t, xchim, deriv: pression, température, composition chimique par gramme, deriv=.TRUE. calcul des dérivées;
- Sorties :
  - ro, drop, drot, drox: densité et dérivées;
  - u, dup, dut, dux: énergie interne et dérivées;
  - delta, deltap, deltat, deltax: δ et dérivées;
  - cp,dcpp,dcpt,dcpx: $c_{\rm p}$ et dérivées;
  - gradad, dgradadp, dgradadt, dgradadx: $\nabla_{\rm ad}$ et dérivées;
  - alfa, beta, gamma1: α, β, Γ_1.

7.25 Submodule submod_etat_ceff

7.25.1 Routine etat_ceff

La routine PRIVATE etat_ceff du module mod_etat, *cf.* §D.5 (Page 363), utilise le formalisme CEFF. C'est une adaptation à Cesam2k20 des programmes originaux de J.Christensen-Dalsgaard par A.Baglin, M.Auvergne, Y. Lebreton, P.Morel & B.Pichon.

Les routines qui relèvent de ce package sont les suivantes :

bilinc, clmnew, cvmgp, cvmgz, cvmgn, cvmgt, cvmgm, dmpeqs, derive, eoststst, eqstfc, eqstpc, eqstrh, eosder, f4der, f4n, f4mhd, hheion, hmnion, hvionac, inteffc, ismax, isamax, ismin, leq, neweta, omegac, phderc, prthvi, setgm1, setf4, seteqs, blockdata bleqstc, setcnsc, sdot, ssum, sscal, scopy, saxpy, storec, zeroc.

Ces routines sont regroupées dans un même package et compilées simultanément avec etat_ceff; elles sont donc des routines PRIVATE du module mod_etat, *cf.* §D.5 (Page 363). Diverses modernisations concernant la programmation ont été effectuées par B.Pichon et P.Morel. Le BLOCKDATA initial a été transformé en un tableau de paramètres. D'origine externe, ces routines ne sont pas détaillées.

Problem: *Cette routine qui demande beaucoup de ressources est peu robuste, son usage est délicat.*

7.26 Submodule submod_etat_eff

7.26.1 Routine etat_eff

La routine PRIVATE etat_eff du module mod_etat, *cf.* §D.5 (Page 363), utilise le formalisme EFF, la source, rassemblée dans le package EFF, est une adaptation à Cesam2k20 des programmes originaux de J. Christensen-Dalsgaard par A. Baglin, M. Auvergne & P.Morel.

Les routines qui relèvent de ce package sont les suivantes :

bilin, eqstf, eqstp, hviona, inteff, omega, phder, setcns, store, theffp, zero.

Elles sont exploitées sous forme de routines subordonnées de la routine etat_eff. Le BLOCKDATA initial a été transformé en un tableau de paramètres.

7.27 Submodule submod_etat_gong

7.27.1 Routine etat_gong1

Selon les prescriptions du "Solar Model Comparison Project" Christensen-Dalsgaard (1988), la routine PRIVATE etat_gong1 du module mod_etat, *cf.* §D.5 (Page 363), suppose le milieu totalement ionisé, mais ne tient compte ni de la dégénérescence, ni de l'énergie interne d'ionisation, ni de la pression de radiation.

7.27.2 Routine etat_gong2

Suivant "Solar Model Comparison Project" Christensen-Dalsgaard (1988), la routine PRIVATE etat_gong2 du module mod_etat, *cf.* §D.5 (Page 363), utilise le formalisme EFF simplifié, sans tenir compte, ni de la pression de radiation, ni de la dégénérescence. La mise en oeuvre de etat_gong2 a bénéficié de la collaboration de J. Provost.

7.28 Submodule submod_etat_mhd

7.28.1 Routine etat_mhd

La routine PRIVATE etat_mhd du module mod_etat, *cf.* §D.5 (Page 363), assure l'interface avec l'équation d'état solaire tabulée MHD de W.Däppen *cf.* **?**. Les tables, en ASCII compressé, sont distribuées dans le package de l'Equation of State MHD, par W.Däppen. Les plus récentes *i.e.* avec "correction τ", sont disponibles à l'adresse FTP anonymous usc.edu directory pub/astro-physics/mhd-oc/wd-evo; leur mise en place est décrite au §**??** (Page **??**).

Les routines qui relèvent de ce package sont regroupées et compilées simultanément avec etat_mhd; ce sont donc des routines PRIVATE du module mod_etat. Il s'agit de :

fmttob, mhdpx, mhdpx1, mhdpx2, intpt, mhdst, mhdst1, rtab, rabu, quint, quintd, lir.

Les noms des tables à utiliser sont indiqués directement dans etat_mhd il n'y a donc pas lieu d'en indiquer les noms dans le fichier de données. Ces tables doivent être créées par l'utilisateur, *cf.* §**??** (Page **??**), et placées dans le sous-directory SUN_STAR_DATA, *cf.* §**??** (Page **??**).

7.29 Submodule submod_etat_opal5Z

7.29.1 Routines etat_opal, etat_opalX, etat_opalZ

Ces routines PRIVATE du module mod_etat, *cf.* §D.5 (Page 363), assurent l'interface avec les diverses versions de l'équation d'état tabulée OPAL (**?**). Les tables d'Equation of State, disponibles sur le site WEB ftp://www-phys.llnl.gov/pub/opal/eos/ sont très grandes.

Suivant les valeurs de l'abondance en métaux requise, il faut contruire les tables ainsi qu'il est décrit au §**??** (Page **??**), et les placer dans le sous-directory SUN_STAR_DATA, *cf.* §**??** (Page **??**). Les routines utilisent les tables OPAL en binaire. La transformation des BLOCKDATA de la distribution sous la forme de paramètres, a permis des gains de place et de temps calcul appréciables. La mise en oeuvre de ces routines a bénéficié de collaborations de S.Brun, J.M. Marques & L.Piau.

Bien qu'identiques, les routines utilisées pour la lecture et l'interpolation des tables utilisent des paramétrages différents. Il s'agit de :

esac, gmass, quad, r_opal_bin, rad_sub ro,_new, t6rinterp.

Elles sont introduites sous la forme de routines subordonnées et reproduites pour chacune des routines d'équation d'état OPAL.

7.30 Module mod_thermo

7.30.1 Type thermodynamic_var

```
thermodynamic_var: type, public
```

Members:

epsilon : **real**(kind=dp), **dimension**(6)
Nuclear energy [cgs]
epsilon(1) : total energy; epsilon(4) : energy from $3\alpha + C + O$.
epsilon(2) : energy from PP chain; epsilon(5) : Gravitational energy.
epsilon(3) : energy from CNO cycle; epsilon(6) : neutrino energy.

alfa, beta : **real**(kind=dp)

cp : **real**(kind=dp)
Specific heat capacity at constant pressure c_p.

dcpp, dcpt, dcpx : **real**(kind=dp)
Derivatives of c_p wrt pressure, temperature and hydrogene abundance.

delta, deltap, deltat, deltax : **real**(kind=dp)
Derivatives of c_p wrt pressure, temperature and hydrogene abundance.

depsp, depst, depsro : **real**(kind=dp)
énergie nucléaire cgs.

depsx : **real**(kind=dp), **allocatable**, **dimension**(:)
énergie nucléaire cgs.

dgaml, dgamlpp, dgamm, dgamp, dgampt, dgamr, dgamt, dgamx, dgamrs, : **real**(kind=dp)
Convective efficiency and derivatives.

gradrad, gradad, dgradadp, dgradadt, dgradadx : **real**(kind=dp)
Convective efficiency and derivatives.

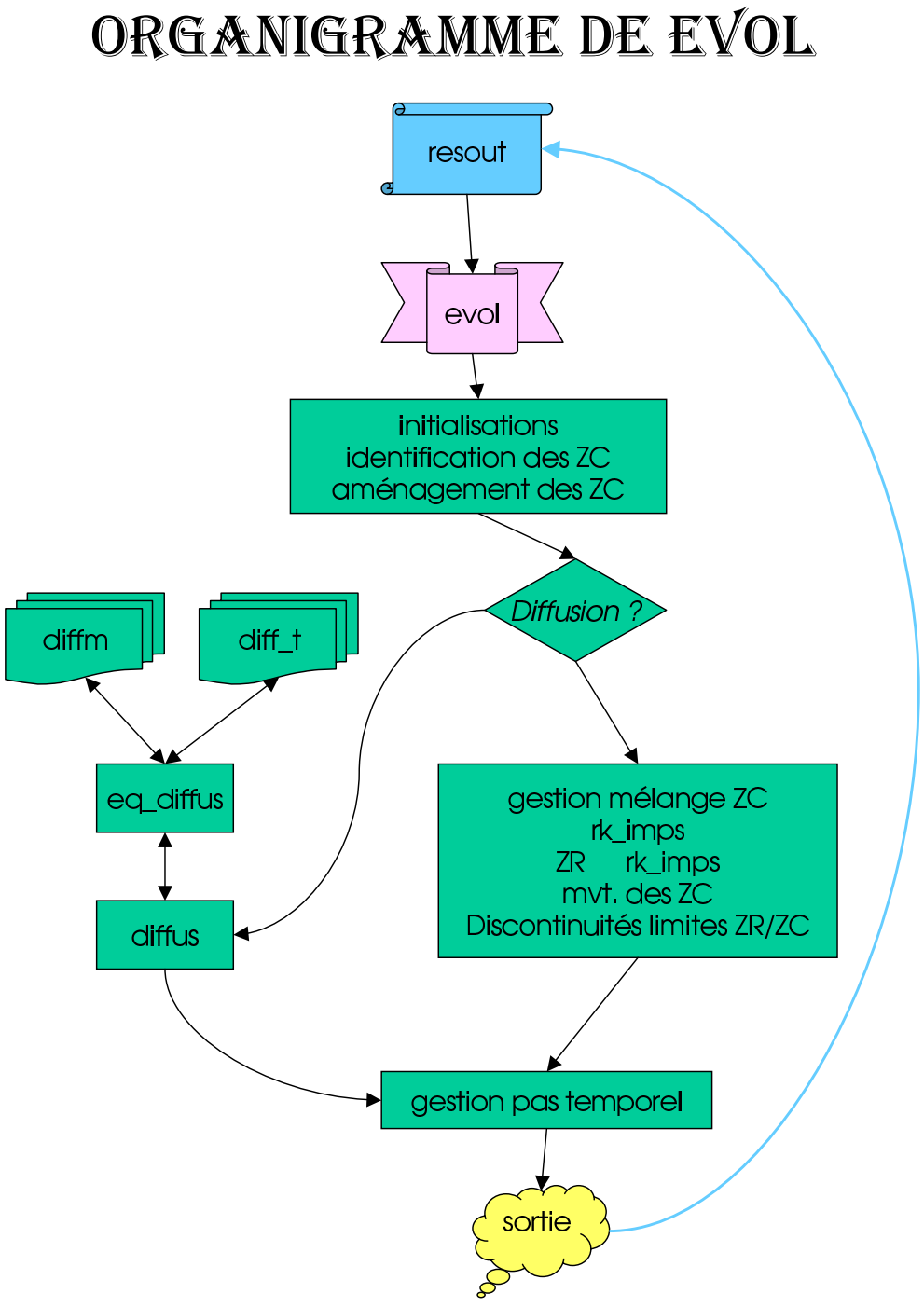


Figure 7.2: Environnement de la routine `evol` de gestion de l'évolution de la composition chimique et du moment cinétique.

grad, dgradl, dgradlpp, dgradm, dgradpt, dgradp, dgradr, dgradrs,dgradt, dgradtau, dgradx : **real**(kind=dp)
Actual gradient ($\nabla \equiv \mathrm{d}\ln T/\mathrm{d}\ln P$) and derivatives.

dgradad_grad, d_grad : **real**(kind=dp)
$\nabla_{\rm ad} - \nabla$.

hp, dhppt, dhpp, dhpt, dhpx, dhpr, dhpm : **real**(kind=dp)
Pressure scale height and derivatives.

kap, dkapp, dkapt, dkapx, dkapro : **real**(kind=dp)
Opacité Rosseland [cm^2/g].

ro, rob, drop, drot, drox, fd : **real**(kind=dp)
Densité [cgs] and derivatives.

u, dup, dut, dux : **real**(kind=dp)
Internal energy and derivatives.

gam : **real**(kind=dp)
Gamma de la convection MLT, sert avec la pression turbulente.

gamma1 : **real**(kind=dp)

grad_mu, dlnmu, mu, grad_mu_shp : **real**(kind=dp)

vconv : **real**(kind=dp)
Convective velocity.

s : **real**(kind=dp)
Entropy (per 10^6 K).

z_bar : **real**(kind=dp), **allocatable**, **dimension**(:)
Average charge per element.

nchim : **integer**
Number of chemical elements followed.

cte1, cte2, cte8, cte9, cte13, cte10, cte11 : **real**(kind=dp)
Number of chemical elements followed.

radiatif : **logical**
If true: radiative zone.

inited : **logical**
If true, class is inited. Initial value: `inited =.false.`.

der : **logical**
Initial value: `der=.false.`.

Subroutine eval_thermo

```
subroutine eval_thermo( this, pt, p, t, m, l, r, dlpp, xchim, dxchim )
```

Private subroutine of `mod_thermo` module.
Method of `thermodynamic_var`.
Compute thermodynamic quantities.

Arguments:

this : **class**(thermodynamic_var), **intent**(inout)

xchim : **real**(kind=dp), **intent**(in), **dimension**(nchim)
Chemical composition (per mole).

dxchim : **real**(kind=dp), **intent**(in), **dimension**(nchim)
dxchim/dν ($\nu = m^{2/3}$) (per mole).

pt : **real**(kind=dp), **intent**(in)
Pression totale Ptot.

p : **real**(kind=dp), **intent**(in)
Pression thermique+radiative Pgas.

t : **real**(kind=dp), **intent**(in)
Température K.

m : **real**(kind=dp), **intent**(in)
Masse/msol.

l : **real**(kind=dp), **intent**(in)
Luminosité/lsol.

r : **real**(kind=dp), **intent**(in)
Rayon / rsol.

dlpp : **real**(kind=dp), **intent**(in)
$\mathrm{d}\ln P_{\rm gas}/\mathrm{d}\ln P_{\rm tot}$.

History:

2018-8 : Adaptation of thermo.f90 (Louis Manchon).

7.30.2 Subroutine get_srhd

```
subroutine get_srhd(  s_rhd, mu  )
```

Routine of mod_thermo Retrieve entropy from R-MHD simulations.

Arguments:

mu : **real**(kind=dp), **intent**(in)
Mean molecular weight.

s_rhd : **real**(kind=dp), **intent**(out)
Entropy of the envelope given by a prescription and corrected if needed.

History:

2021-8 : First implementation (Louis Manchon).

Subroutine get_srhd_ludwig99

```
subroutine get_srhd_ludwig99(  s_rhd, mu  )
```

Private routine of mod_thermo Adiabatic entropy given by the prescription of Ludwig et al. (1999)

References:

L99 : Ludwig et al. (1999).

History:

2021-8 : First implementation (Louis Manchon).

Subroutine `get_srhd_magic13`

```
subroutine get_srhd_magic13( s_rhd, mu )
```

Private routine of `mod_thermo`. Adiabatic entropy given by the prescription of Magic et al. (2013)

References:

M13 : Magic et al. (2013).

History:

2021-8 : First implementation (Louis Manchon).

Subroutine `get_srhd_tanner16`

```
subroutine get_srhd_tanner16( s_rhd, mu )
```

Private routine of `mod_thermo` Adiabatic entropy given by the prescription of Tanner et al. (2016)

References:

T16 : Tanner et al. (2016).

History:

2021-8 : First implementation (Louis Manchon).

7.31 Module `mod_evol`

7.31.1 Routine subordonnée `base_chim`

Cette routine subordonnée de `diffus`, génère le vecteur nodal de la base de B-splines utilisée pour la résolution des équations de la diffusion des éléments chimiques, *cf.* § 6.7 (Page 84).

Description :

- Allocation et initialisation du tableau des multiplicités.
- Définition des multiplicités aux limites RZ/CZ.
- Formation du vecteur nodal.

Appel : Le sous-programme `base_chim`, n'a pas d'argument.

7.31.2 Routine subordonnée base_rota

Cette routine subordonnée de diffus et de evol, génère le vecteur nodal de la base de B-splines utilisée pour la résolution des équations de la diffusion du moment cinétique, *cf.* §6.8 (Page 98).

Description :

- Allocation du tableau des multiplicités.
- Initialisation du tableau des multiplicités selon la présence ou l'abscence de limites RZ/CZ.
- Formation du vecteur nodal.

Appel : Le sous-programme base_rota, n'a pas d'argument.

7.31.3 Routine evol

La routine PUBLIC evol du module mod_evol, *cf.* §D.11 (Page 367), gère le mélange convectif, l'intégration temporelle de la composition chimique, éventuellement du moment cinétique. Un organigramme schématique est reproduit Figure 7.2 (Page 179).

Description :

- Initialisations diverses, éliminations éventuelles pour le mélange convectif ***uniquement*** de limites zones radiatives / zones convectives trop externes, détermination si le modèle est totalement radiatif ou convectif.
- Détermination de la répartition en masse - en $(m/M_\odot)^{2/3}$ - utilisée pour l'interpolation de la composition chimique et du moment cinétique. Les points retenus sont, ou bien identiques aux points de la grille utilisée pour les variables globales, ou bien situés aux centres des intervalles entre deux points de cette grille, ou bien encore ceux de la *grille fixe*, *cf.* §3.7 (Page 20). A défaut d'une description physique de l'évolution des discontinuités aux limites zones radiatives / zones convectives l'utilisation alternative des milieux de grille comme points d'interpolation crée une légère diffusion numérique destinée à stabiliser l'évolution. Aux voisinages du centre et des limites zones radiatives / zones convectives, des points supplémentaires sont ajoutés pour améliorer la précision. Cette dernière disposition n'est appliquée qu'avec l'utilisation d'une *grille fixe*.
- Les indices du début et de la fin de chaque zone convective sont ensuite identifiés dans la grille pour l'interpolation de la composition chimique; en enlevant les "c" de commentaires, des algorithmes de test permettent des vérifications.
- Une alternative se présente alors ::
  - On tient compte de la diffusion microscopique avec ou sans diffusion du moment cinétique, il y a appel à la routine diffus, *cf.* §7.17 (Page 171), sauf si le modèle est complètement convectif depuis le début de l'évolution - ou de la reprise du calcul. Au retour de la routine diffus, la précision du calcul de l'évolution des abondances est estimée.
  - Il y intégration et mélange convectif classiques, en tenant compte d'éventuels gains ou pertes de masse, et en estimant la précision du calcul de l'évolution des abondances:
    * d'abord pour les zones convectives où l'intégration de la composition chimique est réalisée de façon simultanée avec le mélange convectif,
    * ensuite pour les zones radiatives.
    * La composition chimique est ensuite ajustée en fonction des retraits ou avancées des limites zones radiatives / zones convectives.

        * Enfin, on calcule les coefficients d'interpolation de la composition chimique en tenant compte des discontinuités aux limites zones radiatives / zones convectives.

- Une renormalisation de la composition chimique de façon à assurer $X+Y+Z=1$ est effectuée dans l'espace des splines. Cette renormalisation est nécessaire pour des arguments:
  - `Physiques:` les masses atomiques n'étant pas des nombres entiers, les équations de l'évolution thermonucléaire de la composition chimique ne peuvent conserver ***exactement*** $X+Y+Z=1$. Pour ce faire, il faudrait tenir compte de corrections relativistes.
  - `Numériques:` les équations implicites de l'évolution de la composition chimique étant résolues de façon itérative, il y a nécessairement une erreur numérique résultant de l'appréciation de la convergence. Les erreurs d'arrondi sont aussi à prendre en ligne de compte.
- Une estimation du pas temporel à venir est faite en fonction des valeurs, requise et obtenue, de la précision de l'intégration. En cas de non convergence d'algorithmes itératifs, ou si la précision requise n'est pas atteinte, le modèle est réinitialisé lors du retour à `resout`, et le calcul est alors repris avec un pas temporel divisé par 2. ***L'estimation du pas temporel ne porte que sur l'estimation de la précision de l'intégration de la composition chimique.***

Appel :
`evol` est appelée par `resout`, *cf.* §7.67 (Page 219).

```
SUBROUTINE evol(compt,dt,dts,reprend)
```

- Entrées :
  - `compt`: compteur des itérations globales, compt=0 pour la première itération globale de chaque nouveau pas temporel.
  - `dt`: pas temporel.
- Sorties :
  - `dts`: estimation du pas temporel à utiliser pour le pas temporel suivant.
  - `reprend=.TRUE.`: il faut réduire le pas temporel à cause de TdS ou de non convergence.

7.31.4 Routine générique `coeff_rota`

Cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), est la gestion du calcul des coefficients de la diffusion du moment cinétique.

Description :
Le calcul est orienté vers la routine définie dans la routine `lit_nl.f90`, *cf.* §7.46 (Page 207), suivant le nom du formalisme à utiliser, *cf.* §?? (Page ??).

Appel :
`coeff_rota` est appelée par `eq_diff_rota`, *cf.* §7.23 (Page 175).

```
  SUBROUTINE coeff_rota(dt,nu,y,frl,ddfrl)
```

- Entrées :
  - `dt, nu, y`: pas temporel, abscisse, tableau des variables de la rotation.

- Sorties :
  - frl, ddfrl: coefficients des équations de la rotation et dérivées par rapport aux variables de la rotation.

7.31.5 Routines coeff_rota3/4

Ces routines ont pour fonction le calcul des coefficients des équations de la diffusion du moment angulaire suivant le formalismes de **??** respectivement.

Description :

- Interpolation au point de calcul des variables de l'équilibre quasi-statique et calcul de divers coefficients.
- Interpolation de la composition chimique, appel à l'Equation of State.
- Estimation des variables de la rotation et de l'équilibre quasi-statique au pas temporel précédent.
- Calcul de l'opacité, des taux d'ionisation, de l'énergie nucléaire et calcul de divers coefficients.
- Interpolation au point de calcul des variables de la rotation.
- Calcul des coefficients de diffusion et des pertes de moment cinétique.
- Calcul des coefficients non identiquement nuls.

Appel :

coeff_rota3/4 sont appelées par coeff_rota eq_diff_rota3/4. Les arguments sont ceux de coeff_rota.

7.31.6 Routine collision

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), calcule les intégrales de collision $\Omega_{ij}^{(kl)},\ k,l=1,2$ en utilisant les tabulations de **?**. Ces tables ont été scannées de l'ApJS. Les quantités qui figurent dans les data sont les valeurs des intégrales de collision calculées par les formules d'interpolation pour les $\psi_{st}(n)$ (notations de **?**), et prolongées jusqu'à $\psi_{st}=4$; au delà de $\psi_{st}=4$, on utilise les approximations analytiques.

Au premier appel de la routine, on calcule les coefficients d'interpolation (appel à bsp1dn, *cf.* §8.2.2 (Page 241)) et on effectue quelques autres initialisations. De la longueur de Debye λ, on déduit les valeurs des ψ_{ij} d'où l'interpolation des intégrales de collision et le calcul de leurs dérivées par rapport à chacune des abondances; distinction est faite entre potentiels attractifs et répulsifs. On économise des ressources de temps calcul en exploitant les symétries.

Appel : collision est appelée par la routine diffm_br *cf.* §7.31.15 (Page 189).

```
1 SUBROUTINE collision(nb,zb,mij,xi,ro,drox,t,omega11,zij,zpij,zsij,
2    1   domega11,dzij,dzpij,dzsij)
```

- Entrées :
  - nb, zb, mij, xi: nombre de particules (l: indice de ^{1}H, nb: indice pour les électrons), charges, masses réduites, abondances par mole.
  - ro, drox, t: densité, dérivée logarithmique par mole, température.
- Sorties :
  - omega11, zij, zpij, zsij: tables des intégrales de collision.
  - domega11, dzij, dzpij, dzsij: tables des dérivées des intégrales de collision.

7.31.7 Routine coulomb

Cette routine PRIVATE du module mod_evol effectue le calcul du logarithme de Coulomb et de ses dérivées par rapport à la densité et à l'abondance en ^{1}H ou ^{4}He, suivant le formalisme simplifié de ?, p. 250.

Appel :
coulomb est appelée par la routine diffm_mp.

```
SUBROUTINE coulomb(zi,zj,thetae,ro,x,t,lnlambij,
     1   lnlambijr,lnlambijx,cij,cijr,cijx)
```

- Entrées :
  - zi, zj, thetae: charges des éléments i et j, thetae=1 pour un mélange hélium/hydrogène;
  - ro, x, t: densité, abondance par volume de ^{1}H ou ^{4}He, température.
- Sorties :
  - lnlambij, lnlambijr, lnlambijx, cij, cijr, cijx: logarithmes de Coulomb et dérivées par rapport à ρ et ^{1}H ou ^{4}He.

7.31.8 Routine générique difft

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), est la ***routine générique*** de calcul de la diffusion turbulente.

Appel : difft est appelée par eq_diff_chim, *cf.* §7.21 (Page 173).

```
SUBROUTINE difft(melange,t,ro,drox,kap,dkapx,deff,gradad,gradrad,m,d,dd)
```

- Entrées :
  - melange: melange=.TRUE. on est dans une zone à mélanger,
  - t, ro, drox, kap, dkapx, deff: température, densité, opacité, et dérivées par rapport à l'abondance de ^{1}H **par gramme**, coefficient de diffusion turbulente due à la rotation,
  - gradad, gradrad, m: gradients adiabatique et radiatifs, abscisse de calcul.
- Sorties :
  - d, dd, v, dv: tableaux $d(i,j),\ dd(i,j,k),\ v(i),\ dv(i,k),\ i,j,k = 1,\ldots,n_{\rm elem}$.

7.31.9 Routine difft_gab

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), effectue le calcul des coefficients de diffusion turbulente. Elle exploite une suggestion de M.Gabriel: pour éviter la sédimentation de l'hélium est des éléments lourds dans la partie supérieure de l'enveloppe, il suffit de mélanger la partie de l'enveloppe jusqu'à la fin de la zone d'ionisation de l'hélium, localisée vers 10^6K.

Description :

- Quelques initialisations sont effectuées au premier appel.
- S'il y a mélange, ou si la tempétature est inférieure à 10^6K la diagonale de la matrice de diffusion est affectée du coefficient de mélange convectif $d_M >> 1$ *cf.* §6.9.6 (Page 120).

- Sans mélange, on ajoute aux termes diagonaux de la matrice de diffusion, les coefficients de diffusion turbulente d_turb *cf.* §3.3 (Page 14), $d_{\rm eff}$ *cf.* §6.8.4 (Page 102).

```
SUBROUTINE difft_gab(melange,t,deff,d)
```

- Entrées :
  - melange: melange=.TRUE. on est dans une zone à mélanger,
  - t,deff: température, coefficient de diffusion turbulente due à la rotation,
- Sortie :
  - d: tableau $d(i,j)$.

7.31.10 Routine difft_nu

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), effectue le calcul des coefficients de diffusion turbulente.

Description :

- Quelques initialisations sont effectuées au premier appel.
- En cas de mélange, la diagonale de la matrice de diffusion est affectée du coefficient de mélange convectif $d_M >> 1$ *cf.* §6.9.6 (Page 120).
- Sans mélange, on ajoute aux termes diagonaux de la matrice de diffusion, les coefficients de diffusion turbulente d_turb *cf.* §3.3 (Page 14), $d_{\rm eff}$ *cf.* §6.8.4 (Page 102) et de viscosité radiative Re_ν, *cf.* **?**.

Appel :
difft_nu, est appelée par difft :

```
SUBROUTINE difft(melange,t,ro,drox,kap,dkapx,deff,d,dd)
```

- Entrées :
  - melange: melange=.TRUE. on est dans une zone à mélanger,
  - t, ro, drox, kap, dkapx, deff: température, densité, opacité, et dérivées par rapport à l'abondance de ^{1}H **par gramme**, coefficient de diffusion turbulente due à la rotation,
- Sorties :
  - d, dd: tableaux $d(i,j),\ dd(i,j,k),\ i,j,k=1,\ldots,n_{\rm elem}$.

7.31.11 Routine difft_sun

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), calcule un coefficient de diffusion turbulente, sous la ZC externe solaire, suivant Gabriel (1997). **Appel :**
difft_sun est appelée par eq_difft,

```
SUBROUTINE difft_sun(melange,deff,gradad,gradrad,m,d)
```

- Entrées :
  - `melange`: melange=.TRUE. on est dans une zone à mélanger,
  - `deff, gradad, gradrad, m`: coefficient de diffusion turbulente due à la rotation, gradients adiabatique et radiatifs, abscisse de calcul.
- Sorties :
  - `d`: tableau $d(i,j)$ $i, j, \ldots, n_{\rm elem}$.

7.31.12 Routine générique `diffw`

Cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), est la ***routine générique*** de calcul des coefficients de diffusion turbulente horizontale, verticale et effective du moment cinétique, *cf.* §6.8 (Page 98). Dans la présente version de Cesam2k20 elle ne gère que la description `diffw_mpz`.

Appel :
`diffw` est appelée par `eq_diff_rota`, *cf.* §7.23 (Page 175).

Arguments de `diffw`

```
SUBROUTINE diffw(dlnro,grand_k,nu,n2mu,n2t,r,ro,dfrot,frot,
    1 deff,dh,dv)
```

- Entrées :
  - `dlnro,grand_k,nu,n2mu,n2t`: gradient $\ln\rho$, K, ν, variables liées à la fréquence de $\mathbf{v}$, *cf.* §6.8.4 (Page 102).
  - `r,ro,dfrot,frot`: rayon, densité, vecteur des variables de la rotation et dérivées.
- Sorties :
  - `deff,dh,dv`: coefficients de diffusion turbulente effective, horizontale, verticale.

7.31.13 Routine `diffw_mpz/p03`

Ces routines PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), effectue le calcul des coefficients de diffusion horizontale, verticale et effective du moment cinétique, selon les prescriptions de **??** respectivement.

Description de `diffw_mpz` :

- Au premier appel, initialisation et diverses écritures.
- Calcul de $D_{\rm h}$
- Calcul de $D_{\rm v}$
- Calcul de $D_{\rm eff}$

Appel :
`diffw_mpz/p03`, appelées par `diffw`, utilisent la même liste d'appel.

7.31.14 Routine générique diffm

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), est la ***routine générique*** de calcul de la diffusion microscopique. Elle gère les descriptions diffm_br, diffm_mp et diffm_0, cette dernière, formelle, fixe à zéro les coefficients de diffusion microscopique.

Description :

Pour chaque élément du vecteur de composition chimique, l'équation de diffusion, avec les termes nucléaires, est formulée dans la routine eq_dif_chim. Les coefficients de diffusion microscopique sont calculés dans les routines de type diffm, et des coefficients de diffusion turbulente et de mélange (moment angulaire inclus) sont introduits par l'intermédiaire des routines de type difft *cf.* §7.31.8 (Page 186).

Appel :

diffm est appelée par eq_diff_chim, *cf.* §7.21 (Page 173).

```
SUBROUTINE diffm(p,t,r,l,m,ro,drox,kap,dkapx,w,gradrad,dgradradx,xi,d,dd,v,dv)
```

- Entrées :
  - p, t, r, lum, ltot, m: pression, température, rayon, luminosité locale et totale, masse.
  - ro, drox, kap, dkapx: densité, dérivée/ X, opacité, dérivée / X,
  - gradrad, dgradradx, xi: gradient radiatif, dérivée / X. composition chimique par mole.
- Sorties :
  - d, dd: matrice des coefficients de diffusion microscopiques d_{ij} de $\frac{\partial x_j}{\partial m}$ et dérivées $\frac{\partial d_{ij}}{\partial x_k}$.
  - v, dv: vecteur des vitesses v_i des x_i et dérivées $\frac{\partial v_i}{\partial x_k}$.

7.31.15 Routine diffm_br

La routine PRIVATE diffm_br du module mod_evol, *cf.* §D.11 (Page 367), permet de calculer les coefficients de diffusion microscopique en résolvant les équations de transport selon le formalisme de Burgers, en tenant compte, éventuellement, des accélérations radiatives *cf.* §7.8.1 (Page 160). La formulation du système des équations de Burgers utilisée dans Cesam2k20, est en grande partie originale. Elle est décrite §6.7.3 (Page 87).

Description :

- Au premier appel, initialisation de constantes, des masses réduites, allocation diverses, initialisation des accélérations radiatives.
- Calcul des taux d'ionisation, calcul du poids moléculaire moyen, de la gravité corrigée des accélérations radiatives et de leurs dérivées.
- Calcul des intégrales de collision, des coefficients de résistance Eq. 6.154 (Page 91) et de leurs dérivées.
- Calcul des coefficients $q_{\bar{ij}}$ Eq. 6.181 (Page 95), d_i Eq. 6.182 (Page 95), f_{ij} Eq. 6.183 (Page 95) et de leurs dérivées.
- Formation des matrices Δ Eq. 6.176 (Page 94), $\mathcal{M}$ Eq. 6.177 (Page 94), A_w Eq. 6.180 (Page 94), A_r Eq. 6.184 (Page 95), A_c Eq. 6.186 (Page 95), A_e Eq. 6.185 (Page 95).
- Calcul de ω par inversion du système linéaire, Eq. 6.187 (Page 95).

- Formation de la matrice des coefficients de diffusion Eq. 6.196 (Page 96), et du jacobien Eq. 6.202 (Page 96).
- Calcul du vecteur des vitesses de diffusion Eq. 6.189 (Page 95) et du jacobien Eq. 6.199 (Page 96).

7.31.16 Routine `diffm_mp`

La routine PRIVATE `diffm_mp` du module `mod_evol`, *cf.* § D.11 (Page 367), utilise la formulation simplifiée de Michaud & Proffitt (1993) pour l'estimation des coefficients de diffusion microscopique. On suppose que les éléments lourds sont *éléments traces*, *i.e.* un élément autre que ^{1}H et ^{4}He ne diffuse que par rapport aux protons. Ce formalisme, plus simple mais moins général que celui de Burgers, **ne s'applique pas** dans un milieu privé de noyaux d'hygrogène, *i.e.* un coeur d'hélium.

Description :

- Au premier appel, initialisation de constantes, des masses réduites, allocations diverses.
- Calcul des logarithmes de Coulomb pour ^{1}H et ^{4}He.
- Calcul des coefficients de diffusion de ^{1}H et de ^{4}He et dérivées.
- Calcul des coefficients de diffusion pour les éléments tests par rapport à ^{1}H et dérivées.

7.31.17 Subroutine `diffmg_ts`

```
subroutine diffmg_ts( nu, dlnro, grand_k, n2mu, n2t, r, ro, eta, y, nu_ts, d_ts, iii,
↪  coeff, kh_valid_t )
```

Public subroutine of `mod_evol` module Generic subroutine for the computation of turbulent diffusion coefficients for rotation and chemical elements

Arguments:

`y` : **`real`**`(kind=dp), intent(in), dimension(2,0:1)`
Rotation quantities
`y(1, 0)` : Ω; `y(1, 1)` : $\partial\Omega/\partial\nu$.
`y(2, 0)` : U; `y(2, 1)` : $\partial U/\partial\nu$.

`dlnro` : **`real`**`(kind=dp), intent(in)`
$\mathrm{d}\ln\rho/\mathrm{d}\nu$.

`grand_k` : **`real`**`(kind=dp), intent(in)`
Thermal diffusivity.

`nu` : **`real`**`(kind=dp), intent(in)`
Mass coordinate.

`n2mu` : **`real`**`(kind=dp), intent(in)`
Chemical part N^2_μ of the Brunt-Vaissala frequency.

`n2t` : **`real`**`(kind=dp), intent(in)`
Thermal part N^2_T of the Brunt-Vaissala frequency.

`r` : **`real`**`(kind=dp), intent(in)`
Radial coordinate.

`ro` : **`real`**`(kind=dp), intent(in)`
Density.

eta : **real**(kind=dp), **intent**(in)
Molecular magnetic diffusivity.

nu_ts : **real**(kind=dp), **intent**(out)
Tayler-Spruit induced viscosity.

d_ts : **real**(kind=dp), **intent**(out)
Magnetic diffusivity.

coeff : **real**(kind=dp), **intent**(out), **dimension**(:), **optional**
Output that contains radial and azimuthal component of the magnetic field.

kh_valid_t : **logical**, **intent**(in), **optional**
True if criterion for Kelvin-Helmoltz instability was valid as same place at previous time step.

iii : **integer**, **intent**(in)
True if criterion for Kelvin-Helmoltz instability was valid as same place at previous time step.

Subroutine diffmg_ts_spruit2002

```
subroutine diffmg_ts_spruit2002( nu, grand_k, n2mu, n2t, r, ro, y, nu_ts, d_ts, coeff,
↪  valid, l_ts )
```

Private subroutine of mod_evol module.
Subroutine for the computation of turbulent diffusion coefficients for rotation and chemical elements induced by the Tayler Spruit instability. This routine follows the initial prescription of Spruit 2002.

Arguments:

y : **real**(kind=dp), **intent**(in), **dimension**(2,0:1)
Rotation quantities
y(1, 0) : Ω; y(1, 1) : $\partial\Omega/\partial\nu$.
y(2, 0) : U; y(2, 1) : $\partial U/\partial\nu$.

grand_k : **real**(kind=dp), **intent**(in)
Thermal diffusivity.

nu : **real**(kind=dp), **intent**(in)
Mass coordinate.

n2mu : **real**(kind=dp), **intent**(in)
Chemical part N^2_μ of the Brunt-Vaissala frequency.

n2t : **real**(kind=dp), **intent**(in)
Thermal part N^2_T of the Brunt-Vaissala frequency.

r : **real**(kind=dp), **intent**(in)
Radial coordinate.

ro : **real**(kind=dp), **intent**(in)
Density.

nu_ts : **real**(kind=dp), **intent**(out)
Tayler-Spruit induced viscosity.

d_ts : **real**(kind=dp), **intent**(out)
Magnetic diffusivity.

coeff : **real**(kind=dp), **intent**(out), **dimension**(2)
Output that contains radial and azimuthal component of the magnetic field.

l_ts : **real**(kind=dp), **intent**(out), **optional**
Tayler's scale: maximum length on which a TS instability can occur.

valid : **logical**, **intent**(out)
True if criterion for Tauler-Spruit instability is valid.

Subroutine diffmg_ts_maeder2004

```
subroutine diffmg_ts_maeder2004( nu, grand_k, n2mu, n2t, r, ro, eta, y, nu_ts, eta_ts,
↪  coeff, valid, iii )
```

Private subroutine of mod_evol module.

Subroutine for the computation of turbulent diffusion coefficients for rotation and chemical elements induced by the Tayler Spruit instability. This routine follows the prescription of Maeder et al. 2004.

Arguments:

y : **real**(kind=dp), **intent**(in), **dimension**(2,0:1)
Rotation quantities
y(1, 0) : Ω; y(1, 1) : $\partial\Omega/\partial\nu$.
y(2, 0) : U; y(2, 1) : $\partial U/\partial\nu$.

grand_k : **real**(kind=dp), **intent**(in)
Thermal diffusivity.

nu : **real**(kind=dp), **intent**(in)
Mass coordinate.

n2mu : **real**(kind=dp), **intent**(in)
Chemical part N^2_μ of the Brunt-Vaissala frequency.

n2t : **real**(kind=dp), **intent**(in)
Thermal part N^2_T of the Brunt-Vaissala frequency.

r : **real**(kind=dp), **intent**(in)
Radial coordinate.

ro : **real**(kind=dp), **intent**(in)
Density.

eta : **real**(kind=dp), **intent**(in)
Molecular magnetic diffusivity.

nu_ts : **real**(kind=dp), **intent**(out)
Tayler-Spruit induced viscosity.

eta_ts : **real**(kind=dp), **intent**(out)
Turbulent magnetic diffusivity.

coeff : **real**(kind=dp), **intent**(out), **dimension**(3)
Output that contains radial and azimuthal component of the magnetic field.

valid : **logical**, **intent**(out)
True if criterion for Tauler-Spruit instability is valid.

iii : **integer**, **intent**(in)
True if criterion for Tauler-Spruit instability is valid.

Subroutine diffmg_ts_daniel2023

```
subroutine diffmg_ts_daniel2023( nu, grand_k, n2mu, n2t, r, ro, y, nu_ts, d_ts, coeff,
↪  valid, l_ts )
```

Private subroutine of mod_evol module.

Subroutine for the computation of turbulent diffusion coefficients for rotation and chemical elements induced by the Tayler Spruit instability. This routine follows the prescription of Daniel et al. 2023.

Arguments:

y : **real**(kind=dp), **intent**(in), **dimension**(2,0:1)
Rotation quantities
y(1, 0) : Ω; y(1, 1) : $\partial\Omega/\partial\nu$.
y(2, 0) : U; y(2, 1) : $\partial U/\partial\nu$.

grand_k : **real**(kind=dp), **intent**(in)
Thermal diffusivity.

nu : **real**(kind=dp), **intent**(in)
Mass coordinate.

n2mu : **real**(kind=dp), **intent**(in)
Chemical part N^2_μ of the Brunt-Vaissala frequency.

n2t : **real**(kind=dp), **intent**(in)
Thermal part N^2_T of the Brunt-Vaissala frequency.

r : **real**(kind=dp), **intent**(in)
Radial coordinate.

ro : **real**(kind=dp), **intent**(in)
Density.

nu_ts : **real**(kind=dp), **intent**(out)
Tayler-Spruit induced viscosity.

d_ts : **real**(kind=dp), **intent**(out)
Magnetic diffusivity.

coeff : **real**(kind=dp), **intent**(out), **dimension**(2)
Output that contains radial and azimuthal component of the magnetic field.

l_ts : **real**(kind=dp), **intent**(out), **optional**
Tayler's scale: maximum length on which a TS instability can occur.

valid : **logical**, **intent**(out)
True if criterion for Tauler-Spruit instability is valid.

7.31.18 Subroutine smoothing_ts

```
subroutine smoothing_ts( r, l_ts, n, nu_ts )
```

Subroutine part of the mod_evol module

Smooth viscosity coefficient induced by the Tayler-Spruit instability. The smoothing is performed with a sliding average over a characteristic lengthscale l_r. The distribution used can be either 'dirac' (i.E. No average, we only take the value computed at a given layer), or 'parabola' (i.E. A 2nd order polynomial with compact support between $c - l_r$ and $c + l_r$).

Arguments:

`r` : `real(kind=dp), dimension(:), intent(in)`
Radius of each layer.

`l_ts` : `real(kind=dp), dimension(:), intent(in)`
Characteristic lengthscale.

`nu_ts` : `real(kind=dp), dimension(:), intent(inout)`
Values of Tayler-Spruit induced viscosity coefficients computed locally (`in`) or averaged (`out`).

`n` : `integer, intent(in)`
Size of arrays.

Orignal author(s):

- L. Manchon.

7.31.19 Subroutine ovshts_diff

```
subroutine ovshts_diff( hp, r, d )
```

Private subroutine of `mod_evol` module.
Computes turbulent diffusion coefficient for diffusive core overshoot following Herwig 2000

Arguments:

`hp, r` : `real(kind=dp), intent(in)`

`d` : `real(kind=dp), intent(inout), dimension(:,:)`
Diffusion coefficient matrix.

History:

2024-2 : First version (Morgan Deal).

7.31.20 Subroutine difft_ovs

```
subroutine difft_ovs( hp, r, d )
```

Private subroutine of `mod_evol` module.
Options for turbulent diffusion coefficient for diffusive core overshoot

Arguments:

`hp, r` : `real(kind=dp), intent(in)`

`d` : `real(kind=dp), intent(inout), dimension(:,:)`
Diffusion coefficient matrix.

History:

2024-2 : First version (Morgan Deal).

7.31.21 Routine générique f_rad

La routine PRIVATE f_rad du module mod_evol, *cf.* §D.11 (Page 367), constitue la ***routine générique*** du calcul des accélérations radiatives. Le calcul de l'accélération radiative est effectué avec la routine dont le nom NOM_FRAD est indiqué dans le fichier de données, *cf.* §3.9 (Page 23).

Appel :
f_rad est appelée par diffm_br, *cf.* §7.31.15 (Page 189).

```
SUBROUTINE f_rad(lum,ray,t,kap,dkapx,nel,ychim,ioni,grav,g_rad,dg_rad)
```

- Entrées :
  - lum, ray, t, kap, dkapx: luminosité, rayon, température, opacité, dérivée/X (mole).
  - nel, ychim, ioni, grav: nombre d'électrons libres par volume, composition chimique par mole, taux d'ionisation, gravité.
- Entrées/Sorties: g_rad(i): vecteur des accélérations radiatives, la gravité est grav+g_rad sur l'élément d'indice i
- Sorties: dg_rad(i,j): matrice des dérivées des accélérations radiatives sur l'élément i par rapport à l'abondance par mole de l'élément j.

7.31.22 Routine générique pertw

Cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), est la ***routine générique*** du calcul de la perte/gain de moment cinétique.

Description :
Suivant la description de la perte de moment cinétique indiquée par la variable NOM_PERTW du fichier de données mon_modele.don, *cf.* §3.3 (Page 14), la routine spécifique de perte de moment cinétique est utilisée.

Appel :
pertw est appelée par coeff_rota, *cf.* §7.31.4 (Page 184).

```
SUBROUTINE pertw(nu,omega,r,mw_dot)
```

- Entrées :
  - nu, omega, r: masse, vitesse angulaire, rayon au point de calcul.
- Sortie
  - mw_dot: taux de perte/gain de moment cinétique.

Routine pertw_loc

Avec cette routine PRIVATE du module mod_evol, *cf.* §D.11 (Page 367), de type pertw, la perte / gain de moment cinétique est proportionnelle à l'énergie cinétique de rotation locale et supposée localisée dans la fraction de masse externe M > pw_extend, *cf.* §6.8.10 (Page 116).

Description :
La variation temporelle de moment cinétique par unité de masse est modélisée par :

$$\dot{\mathcal{M}}_\Omega = \gamma R^2 \Omega^2, \quad |\gamma| \sim 10^{-13} \sim 10^{-14} \tag{7.1}$$

Le paramètre γ correspond au taux `p_pertw` indiqué dans le fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14), le taux de perte/gain de moment cinétique `mw_dot` est nul en deça de la limite fixée.

Appel :

Cette routine utilise les mêmes arguments que sa routine générique `pertw`, *cf.* §7.31.22 (Page 195).

Routine `pertw_ptm`

Avec cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), de type `pertw`, le taux `p_pertw` de perte de moment cinétique indiqué dans le fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14), est utilisé comme un paramètre fixant l'efficacité de la perte de moment cinétique résultant de la perte de masse, *cf.* §6.8.10 (Page 116). La perte de moment cinétique est supposée localisée dans la fraction de masse externe `M > pw_extend`.

Description :

L'apport/perte de moment cinétique par unité de masse est approché par :

$$\dot{\mathcal{M}}_\Omega = \frac{\dot{\mathcal{M}}}{m_{\rm rot}} R^2 \Omega_{\rm p}. \tag{7.2}$$

Le taux de perte/gain de moment cinétique `mw_dot` est fixé à 0 en deça de la limite fixée.

Appel :

Cette routine utilise les mêmes arguments que sa routine générique `pertw`, *cf.* §7.31.22 (Page 195).

Routine `pertw_sch`

Avec cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), de type `pertw`, le taux `p_pertw` de perte de moment cinétique indiqué dans le fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14), est utilisé comme un paramètre fixant l'efficacité d'une perte de moment cinétique proportionnelle à Ω^3 (Schumanish), *cf.* §6.8.10 (Page 116).

Description :

La variation temporelle de moment cinétique par unité de masse est modélisée par :

$$\dot{\mathcal{M}}_\Omega = a R^2 \Omega_{\rm s}^3, \quad |a| \sim 1.10^{-9}{\rm s}. \tag{7.3}$$

Le taux de perte de moment cinétique `mw_dot` est fixé à 0 en deça de la limite fixée.

Appel :

Cette routine utilise les mêmes arguments que sa routine générique `pertw`, *cf.* §7.31.22 (Page 195).

7.32 Submodule `submod_evol2d`

7.32.1 Subroutine `go_to_2D`

```
module subroutine go_to_2D( ok, struct_file )
```

Compute quasi-static structure including centrifugal force in 2D.
Routine of `submod_evol2d`. Implement Roxburgh 2004 & 2006 formalism.

Arguments:

`ok` : `logical`, `intent(out)`
: If not of, dt = dt / 2.

`struct_file` : `character(len=*)`, `intent(in)`, `optional`
: Name of input file with structure if needed.

History:

2018-08 : Adaptation of deform.f90 (Louis Manchon).

2018-09 : Part of submodule evol2d (Louis Manchon).

7.32.2 Subroutine get_geff

```
module subroutine get_geff( n_tot, ok, n_tot_tmp )
```

Retreive g_{eff}-based averages

Compute the average effective gravity using Gauss-Legendre quadrature over an equipotential Private routine of submod_evol2d

Arguments:

n_tot : **integer**, **intent**(in)
Total number of layer: internal structure + atmosphere.

ok : **logical**, **intent**(inout)
If not ok, dt = dt / 2.

n_tot_tmp : **integer**, **intent**(out)
New size of arrays with duplicate layers removed.

History:

2018-8 : Original Code (Louis Manchon).

7.32.3 Subroutine get_geff_general

```
module subroutine get_geff_general(ok, n_tot_tmp)
```

Retreive g_{eff}-based averages

Compute the average effective gravity using Gauss-Legendre quadrature over an equipotential Private routine of submod_evol2d

Arguments:

ok : **logical**, **intent**(inout)
If not ok, dt = dt / 2.

n_tot_tmp : **integer**, **intent**(out)
New size of arrays with duplicate layers removed.

History:

2018-8 : Original Code (Louis Manchon).

7.32.4 Subroutine get_phi

```
module subroutine get_phi( )
```

Compute ϕ including centrifugal force.

Private routine of submod_evol2d

Found gravitational-like potential ϕ from ρ. This graito-rotational "potential" ϕ is computed over sphere from ρ on spheres. The routine get_rho atually computed. The density on isobar but reinterpolated it on spheres at the end. Method is taken from Roxburgh (2006).

Solve following equations:

$\rho_k = W_{kn}^{-1}\rho_n, W_{kn} = P_{2k}(\cos\theta_n)$, with k, $n = 0, N_{\rm leg}$

$$\phi_k = r^{2k}\int_{R_{\rm s}}^{r}\frac{4\pi G}{r^{4k+2}}\left[\rho_k(r)r^{2k+2}\mathrm{d}r\right]\mathrm{d}r - \lambda_k r^{2k}$$

$$\lambda_k = \frac{4\pi G}{(4k+1)R_{\rm s}^{4k+1}}\int_0^{R_{\rm s}}\rho_k(r)r^{2k+2}\mathrm{d}r$$

History:

2018-4 : Translation in FORTRAN (Louis Manchon).

2018-9 : Part of submodule evol2d (Louis Manchon).

7.32.5 Subroutine get_rho

```
module subroutine get_rho( n_tot, ok )
```

Compute ρ including centrifugal force.

Private routine of submod_evol2d Found denity ρ from ϕ. Rho is computed on characteristics (<=> isobars). At the end, we reinterpolate the density on spheres because the routine get_phi will compute the gravito-rotation "potential" over spheres. Method is taken from Roxburgh (2006).

Arguments:

n_tot : **integer**, **intent**(in)
Total number of layer: internal structure + atmosphere.

ok : **logical**, **intent**(inout)
If not of, dt = dt / 2.

History:

2018-4 : Translation in FORTRAN (Louis Manchon).

2018-9 : Part of submodule evol2d (Louis Manchon).

7.32.6 Function get_theta_m

```
module function get_theta_m( rho_char_l ) result (thetam)
```

Retrieve new angle $\theta_{\rm m}$ such that $X(r, \theta_{\rm m}) = X_{\rm 1D}$.

Arguments:

rho_char_l : **real**(kind=dp), **intent**(in), **dimension**(:)
Density array used to determine new angle θ_{m}.

History:

2019-1 : Part of submodule evol2d (Louis Manchon).

7.32.7 Subroutine resout_rota2d

```
module subroutine resout_rota2d(dt, ok, mc_tmp, nc_tmp)
```

Solve advection/diff equations of angular momentum in 2D
public subroutine of submod_evol2d Resolution of the system of non-linear differential equations for the advection/diffusion of angular momentum in 2D.

Arguments:

dt : **real**(kind=dp), **intent**(in)
Time step.

mc_tmp : **real**(kind=dp), **intent**(in), **dimension**(:)
Temporary masses $m^{2/3}$ for interpolation of chemical composition.

nc_tmp : **integer**, **intent**(in)
Size of mc_tmp.

ok : **logical**, **intent**(out)
Convergence ?

History:

2018-12 : Initial Code (Louis Manchon).

7.33 Module mod_static

7.33.1 Routine coll_qs

La fonction de cette routine PRIVATE du module mod_static, *cf.* §D.12 (Page 367), effectue la résolution, dans l'espace des B-splines, du sytème Eq. 6.27 (Page 62) des équations de la structure interne (équilibre quasi-statique).

Description :

- Initialisation du vecteur nodal et des limites, initialisations et allocations diverses lors du premier appel.
- Changement de la base de B-splines si le nombre de couches a varié depuis l'appel précédent.
- Pour chaque point de collocation et chaque point limite, formation par appels à static, *cf.* §7.75 (Page 225), des équations 6.27 (Page 62) dans l'espace des B-splines.
- Résolution du système et évaluation des corrections et de la précision de la méthode itérative, retour à resout ou poursuite des itérations (boucle infinie).

Appel : coll_qs est appelée par resout, *cf.* §7.67 (Page 219).

```
SUBROUTINE coll\_qs(dt,compt,reprend,err,vare,qmax,corr)
```

- Entrées :
  - `dt, compt, reprend=.TRUE.`: pas temporel, compteur du nombre d'itérations, il faut réinitialiser la solution.
- Sorties :
  - `err, vare, qmax, corr`: erreur maximale, variable présentant cette erreur, indice de la couche où l'erreur est maximale, correction.

7.33.2 Routine subordonnée `fcmax`

Cette routine subordonnée de `resout` a pour fonction d'indiquer s'il sera nécessaire d'utiliser le nombre maximum de couches pour les modèles à calculer.

Description :

- Définition de la variable logique `logic` suivant la réalisation d'un des critères d'utilisation du nombre maximum de couches.
- Définition de la variable logique de sortie `cmax` suivant les critères de sortie ASCII définis au §C.1.4 (Page 349).

Appel :
`fcmax` est appelé de `resout` à l'issue de la convergence du modèle.

```
SUBROUTINE fcmax(clogic)
```

- Sortie :
  - `clogic = .TRUE.`: il faudra utiliser le nombre maximum de couches.

7.33.3 Type `dt_control_crit`

Subroutine `check_dtcontrol_crit_agemax`

```
subroutine check_dtcontrol_crit_agemax( this, dt, dts, list_sort, update_dts )
```

Subroutine of `mod_static`.
Method of `dtcontrol_crit`. Checks if age of the model does not exceed maximum age set in `.Don` file.

Arguments:

`this` : `class(dtcontrol_crit)`, `intent(inout)`
`dtcontrol_crit` instance.

`dt` : `real(kind=dp)`, `intent(inout)`
Original time step.

`dts` : `real(kind=dp)`, `intent(inout)`
Counter from `resout`.

list_sort : **logical**, **intent**(inout)
Error from resout.

update_dts : **logical**, **optional**, **intent**(in)
If True, the dts is updated.

7.34 Routine iben

Cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), de type nuc, *cf.* §7.6.1 (Page 149), calcule l'énergie gravifique pour dl'initialisation de l'évolution avec pré-séquence principale, *cf.* §6.4.3 (Page 73). Cette initialisation est effectuée dans la routine cesam, *cf.* §7.7.3 (Page 158); à cet effet, le nom de la routine de réactions thermonucléaires à utiliser est temporairement redéfini: nom_nuc=iben.

Appel :
iben, appelée par la routine générique nuc. Les arguments sont ceux d'une routine de type nuc.

7.35 Routine générique ini_ctes

Cette routine PUBLIC du module mod_donnees, *cf.* §D.3 (Page 353), est la ***routine générique*** pour l'initialisation de la plupart des constantes physiques. Suivant le nom indiqué dans le fichier de données, *cf.* §3.3 (Page 14), il est fait appel à la routine d'initialisation correspondante, *cf.* §**??** (Page **??**).

Appel :
Cette routine générique est appelée par lit_nl, *cf.* §7.46 (Page 207), elle **n'a pas d'argument**.

7.36 Routine initialise_rota

La fonction de cette routine PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), est l'initialisation des fonctions décrivant la diffusion du moment cinétique.

Description :

- Détermination du rayon de l'étoile $R_\star$.
- Appels aux routines w_initial, *cf.* §7.38 (Page 202), et u_initial, *cf.* §7.37 (Page 201), pour l'initialisation de la vitesse angulaire Ω et de la vitesse de circulation méridienne U.
- Déduction des valeurs initiales de θ, Λ et ψ de celles de Ω et U.

Appel :
initialise_rota, appelée par cesam, *cf.* §7.7.3 (Page 158), n'a pas d'argument.

7.37 Fonction initialise_u

Cette fonction PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), initialise la vitesse de circulation méridienne U du formalisme de la diffusion du moment cinétique, *cf.* §6.8.5 (Page 102).

Description :
La valeur initiale de la vitesse de circulation méridienne est initialisée par :

$$|\Omega| U_0 \left(2 - \frac{R}{R_\star)^2}\right)^{-1}, \quad U_0 \sim 0.01. \tag{7.4}$$

Appel :
initialise_u est appelée par initialise_rota, *cf.* §7.36 (Page 201).

```
FUNCTION initialise_u(r,rstar)
```

- `r, rstar`: rayon local, rayon total.

7.38 Fonction `initialise_w`

Cette fonction PRIVATE du module `mod_cesam`, *cf.* §D.13 (Page 368), initialise la vitesse angulaire Ω du formalisme de la diffusion du moment cinétique, *cf.* §6.8.5 (Page 102).

Description :

La valeur locale de la vitesse angulaire est donnée par :

$$\Omega = |\Omega_0| \left\{ 1 + \Omega_f \left[1 + \cos\left(\pi \frac{R}{R_\star} \right) \right] \right\}, \tag{7.5}$$

Ω_0 et Ω_f sont respectivement la vitesse angulaire initiale et le facteur de forme définis dans le fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14).

Appel :

`initialise_w` est appelée par `initialise_rota`, *cf.* §7.36 (Page 201).

```
FUNCTION initialise_w(r,rstar)
```

- `r, rstar`: rayon local, rayon total.

7.39 Routine `inter`

La fonction de cette routine PUBLIC du module `mod_variables`, *cf.* §D.4 (Page 360), est de déterminer par interpolation inverse, à partir de son développement sur une base de B-splines, l'indice de la couche correspondant à une masse ou à un rayon donné. Puis, par interpolation directe, de déterminer en ce point les valeurs des variables et de leurs dérivées premières, par rapport à la variable de masse ou de rayon. En variables lagrangiennes, l'interpolation est effectuée en $(m/M_\odot)^{2/3}$; en variables eulériennes l'interpolation est effectuée en $(r/R_\odot)$. La variable d'indice 6 *i.e.* $f(6)$, affectée à la fonction de répartition, *cf.* §6.2.3 (Page 61), est utilisée pour transmettre l'indice décimal de la couche où s'effectue l'interpolation.

Appel :

`inter` est appelée par divers programmes, en particulier `evol`.

```
SUBROUTINE inter(m23_ou_r2,bp,q,qt n,knot,x,f,dfdx,r2,m23)
```

- Entrées :
  - `m23_ou_r2`: chaîne de ***3 caractères***, si `m23_ou_r2='m23'` interpolation en masse, si `m23_ou_r2='r2␣'` interpolation en rayon.
  - `bp, q, qt, n, knot, x`: éléments pour l'interpolation.
  - `r2, m23`: points d'interpolation.
- Sorties :
  - `f, dfdx`: fonctions ($f(6)$: indice fractionnaire d'interpolation), dérivées.

7.40 Routine inter_atm

Pour l'atmosphère, cette routine PUBLIC du module mod_exploit, *cf.* §D.14 (Page 368), a une fonction similaire à la routine inter précédente. Elle est utilisée par le programme for037_2k du module mod_exploit, *cf.* §?? (Page ??), qui permet l'interpolation d'un modèle en des masses ou des rayons définis à l'avance.

Description :

- Localisation de l'intervalle d'interpolation en masse, rayon ou indice de couche.
- Détermination de l'indice fractionnaire d'interpolation inverse par algorithme de Brent.
- Interpolation des variables de l'atmosphère.
- L'indice fractionnaire est conservé.

Appel :
inter_atm est appelée par le programme for037_2k.

```
1 SUBROUTINE inter_atm(m_ou_r,ne_atm,bp_atm,x_atm,xt_atm,n_atm,
2     1 ord_atm,knot_atm,m_atm,r_atm,x,f,dfdx)
```

- Entrées :
  - m23_ou_r2: chaîne de 3 caractères, interpolation en masse si m23_ou_r2='m23', ou interpolation en rayon si m23_ou_r2='r2␣'.
  - ne_atm, bp_atm, x_atm, xt_atm, n_atm, ord_atm, knot_atm: éléments pour l'interpolation par B-spline.
  - m_atm, r_atm: masses et rayons de la discrétisation de l'atmosphère.
- Sorties :
  - f, dfdx: fonctions, dérivées, rayon, masse.
  - f(ne_atm+1)=x_int: indice fractionnaire de l'abscisse d'interpolation.

7.41 Routine kappa_cond

Cette routine PRIVATE du module mod_opacite, *cf.* §D.6 (Page 364), calcule l'opacité conductive en utilisant les ajustements polynômiaux de ?, et complète l'opacité radiative.

Description :

Bien que remaniée pour son adaptation à Cesam2k20, cette routine, en grande partie d'origine externe, n'est pas détaillée. La moyenne harmonique avec l'opacité radiative est effectuée à l'issue du calcul.

Appel :

Cette routine est appelée par la plupart des routines de calcul d'opacité à l'issue du calcul de l'opacité radiative.

```
1   SUBROUTINE  kappa_cond(xh,t,ro,kappa,dkapdt,dkapdr,dkapdx)
```

- Entrées :
  - xh: composition chimique en fraction de masse.

  - t: température.
  - ro: densité.

- Entrées/Sorties :
  - kap: opacité.
  - dkapdt, dkapdr, dkapdx: dérivées de l'opacité en fonction de la température, densité, abondance d'hydrogène.

7.42 Routine lim_zc

Cette routine PRIVATE du module mod_static, *cf.* §D.12 (Page 367), a trois fonctions principales :

- Initialisation, si nécessaire, de la répartition des couches en fonction du nombre de points à utiliser.
- Localisation des limites des zones radiatives et convectives, au besoin avec overshoot.
- Détermination des coefficients de répartition.

lim_zc est une routine importante de Cesam2k20, son organigramme est présenté Figure 7.1 (Page 167).

Description :

- Initialisation :
  - Initialisations diverses effectuées au premier appel, en particulier, détermination des points limites et calcul des coefficients d'intégration correspondants à cette répartition.
  - Au cas où le nombre de couches doit être modifié, détermination de la nouvelle fonction d'espacement, changement de la base de B-splines, redéfinition des points limites et calcul des coefficients d'intégration. Avec l'option grille_fixe, *cf.* §3.7 (Page 20), la grille d'interpolation de la composition chimique n'est modifiée que si le nombre de couches a varié de plus de la quantité dn_fixe fixée dans resout, *cf.* §7.7.3 (Page 158).
- Localisation :
  - Calcul, en chaque point de la grille, de la différence $\nabla_{\rm rad} - \nabla^*_{\rm ad}$, localisation des changements de signe et identification de la nature de la limite à l'aide de la variable logique lconv qui est .TRUE. lorsqu'on passe d'une zone radiative à une zone convective, en traversant la limite dans le sens croissant de la variable d'espace *i.e.* masse, rayon ou encore, indice.
  - Elimination des zones convectives mal définies *i.e.* trop proches du centre, de la limite externe ou encore n'affectant que très peu de couches.
  - Affinement, par dichotomie, de la localisation des limites retenues.
  - Si besoin, extensions des zones convectives et analyse des chevauchements éventuels.
- Détermination :
  - Des coefficients de répartition qui permettent d'amener chacune des limites sur un point de la grille.

Appel :
lim_zc, appelée par resout, n'a que des arguments d'entrée.

```
SUBROUTINE lim_zc(no_rep,new_nqs)
```

no_rep=.TRUE.: Il n'y a pas eu d'initialisation ou de réinitialisation du modèle.

new_nqs=.TRUE.: Il faut effectuer une redistribution des couches du modèle quasi-statique, la constante de répartition ayant été modifiée.

7.43 Routine list

Cette routine PRIVATE du module mod_cesam, *cf.* §D.13 (Page 368), n'a que des arguments d'entrée. Sa fonction est de construire le listing permettant de suivre le déroulement du calcul et de produire, pour différentes époques de l'évolution, une liste détaillée des variables du modèle, des abondances, des rapports isotopiques à la surface et des flux de neutrinos - l'implantation de ces derniers a bénéficié d'une collaboration avec G.Berthomieu.

Description :

- Initialisations diverses, en particulier des indices d'identification des éléments chimiques.
- Construction d'un cartouche indiquant: l'âge du modèle, la température effective, la luminosité, le rayon, la gravité, la pression, la température, les abondances par fraction de masse de ^{1}H et ^{4}He centrales, les proportions d'énergie libérées par les cycles thermonucléaires et la gravité, la nature du modèle, la variation relative de masse, ses valeurs présente et initiale, les abondances et les rapports isotopiques en surface, éventuellement la période et la vitesse de rotation de surface.
- *Si le listing complet est requis*, le modèle est listé couche par couche du sommet de l'enveloppe vers le centre, un cartouche étant disposé toutes les 10 couches pour faciliter l'identification des quantités écrites. Les limites zones radiatives / zones convectives sont matérialisées. L'atmosphère est ensuite reproduite couche par couche, en partant de l'enveloppe vers la surface; on indique ensuite la déplétion des éléments en surface, les abondances relatives en nombre à la surface, les flux de neutrinos, et une estimation, pour le cas solaire, des grandeurs astérosismologiques: $\nu_0,\ \delta\nu_{02},\ \delta\nu_{13},\ A$.

Appel :
list est appelée par cesam, *cf.* §7.7.3 (Page 158).

```
SUBROUTINE list(alfa,anub8,anube7,anun13,anuo15,anupep,anupp,beta,
     1 compg,cp,delta,dcapdr,dcapdt,depsdr,depsdt,d2p,d2ro,
     2 chaine,convec,ecritout,epsilon,gamma,gamma_atm,gradad,grada_atm,
     3 gradconv,gradc_atm,gradrad,gradr_atm,hp,i_cno,i_gr,i_pp,i_3a,
     4 kap,l,m,mu,mue,m_atm,p,pt,pt_atm,p_atm,r,ro,ro_atm,r_atm,t,tau,
     5 teff,tx_ioni,t_atm,u,vaissala,w,z,degene)
```

- anub8, anube7, anun13, anuo15, anupep, anupp: Tables des nombres de neutrinos des diverses sources.
- alfa, beta, delta: $\alpha,\ \beta,\ \delta$.
- compg, cp, hp: Composition chimique par unité de masse, chaleur spécifique à pression constante, échelle de hauteur de pression.
- kap, dcapdr, dcapdt: Opacité, dérivées par rapport à la densité et à la température.

- `epsilon, depsdr, depsdt`: Energie nucléaire des différents cycles et énergie graviphique, dérivées par rapport à la densité et à la température.
- `d2p, d2ro`: Dérivées secondes de la pression et de la densité centrales.
- `chaine, convec, ecritout`: Statut du modèle, nature radiative ou convective de la couche, écriture du listing complet.
- `gradad, gradconv, gradrad, gamma`: Gradient adiabatique, gradient convectif, gradient radiatif, Γ.
- `grada_atm, gradc_atm, gradr_atm, gamma_atm`: Gradient adiabatique, gradient convectif, gradient radiatif, Γ pour l'atmosphère.
- `i_cno, i_pp, i_3a, i_gr`: Pourcentages d'énergie d'origine nucléaire PP, CNO, 3α et gravifique.
- `mu, mue, vaissala`: Poids moléculaire moyen, poids moléculaire moyen par électron libre, fréquence de $\mathbf{v}$.
- `p, pt, t, r, l, m, ro, u`: Pression gazeuse, pression totale, température, rayon, luminosité, masse, densité, énergie interne spécifique.
- `p_atm, pt_atm, t_atm, r_atm, m_atm, ro_atm`: Pression gazeuse, pression totale, température, rayon, masse, densité dans l'atmosphère.
- `tau, teff`: Epaisseur optique, température effective
- `tx_ioni, w, z, degene`: Taux d'ionisation, vitesse angulaire, abondance des métaux, facteur de dégénérescence.

7.44 Routine `lit_binaire`

Cette routine PUBLIC du module `mod_exploit`, *cf.* §D.14 (Page 368), permet la lecture des fichiers binaires, *cf.* §B (Page 343), à l'exception des fichiers d'atmosphère.

Description :

- Recherche et lecture du fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14).
- Lecture des paramètres du modèle, calcul des dimensions et allocation des tableaux.
- Lecture du modèle.

Appel :

`lit_binaire` appelée par des programmes d'exploitation, *cf.* §?? (Page ??), du module `mod_exploit`, *cf.* §D.14 (Page 368), n'a que des arguments d'entrée.

```
1 SUBROUTINE lit_binaire(chaine,dt)
```

- `chaine, dt`: Nom du modèle, pas temporel.

7.45 Routine lit_hr

Cette routine PUBLIC du module mod_exploit, *cf.* §D.14 (Page 368), permet la lecture des fichiers ASCII de diagramme HR mon_modele.HR, *cf.* §C.2 (Page 349).

Description :

- Lecture du premier enregistrement du fichier ASCII mon_modele.HR, pour allocations.
- Lecture du fichier dans son intégralité.
- Formation des tests de lecture.

Appel :
lit_hr est appelée par des programmes d'exploitation, *cf.* §?? (Page ??), du module mod_exploit, *cf.* §D.14 (Page 368), en particulier par le programme des_hr.

```
SUBROUTINE lit_hr(init,chaine,fin,erreur,log_l,log_r,log_teff,
    1 vrot,wrot)
```

- Entrées :
  - init=.TRUE., chaine: Initialisation, nom du modèle.
- Sorties :
  - fin=.TRUE., erreur=.TRUE.: Fin de fichier, erreur de lecture.
  - log_l, log_r, log_teff : Logarithmes décimaux de la luminosité, du rayon, de la température effective.
  - vrot, wrot: vitesses linéaire et angulaire de la couche externe.

7.46 Routine lit_nl

La fonction de cette routine PUBLIC des modules mod_donnees, *cf.* §D.3 (Page 353) et mod_exploit, *cf.* §D.14 (Page 368), est de lire les NAMELISTS du fichier d'entrée mon_modele.don, *cf.* §3.3 (Page 14).

Description :

- Recherche et lecture des NAMELISTs du fichier de données mon_modele.don, *cf.* §3.3 (Page 14). En cas d'échec de lecture, tentative de trouver et d'utiliser des fichiers de données de versions précédentes de CESAM.
- Vérification de la cohérence des données.
- Détermination des abondance par masse de l'hydrogène X, hélium Y et métaux Z, mise en place de tests.
- Initialisation des principales constantes de physique par appel à la routine ini_ctes, *cf.* §7.35 (Page 201).
- Définition du type de rotation à utiliser.

Appel :
lit_nl est appelée par cesam, *cf.* §7.7.3 (Page 158).

```
SUBROUTINE lit_nl(wrot)
```

- Sortie :
  - `wrot`: Vitesse angulaire.

7.47 Fonction logique `lmix`

La valeur de cette fonction PRIVATE du module `mod_evol` est .TRUE. dans les zones convectives.

Description :

- Un premier test détermine si on se trouve dans la zone convective externe.
- Une fois la localisation de l'abscisse obtenue, on affecte à la fonction la valeur de la table de mélange pour l'indice obtenu.

Appel :

`lmix` est appelée principalement des routines concernées par la diffusion du moment cinétique et des éléments chimiques.

```
    LOGICAL FUNCTION lmix(nu)
```

- `nu`: $(\text{masse})^{\frac{2}{3}}$.

7.48 Routine `modif_mix`

Cette routine PRIVATE du module `mod_nuc`, *cf.* §D.9 (Page 365), permet de personnaliser une mixture, *cf.* §3.15.1 (Page 30).

Description :

- Recherche du fichier de la nouvelle mixture `mon_modele.modif_mix`, *cf.* §3.15.1 (Page 30).
- Lecture des modifications en DeX des abondances initiales.
- Modification des abondances des éléments lourds.

Appel :

`modif_mix`, appelée par `abon_ini`, *cf.* §7.6.1 (Page 148), n'a pas d'argument.

7.49 Routine générique `opa`

La routine PUBLIC `opa` du module `mod_opa`, *cf.* §D.6 (Page 364), est la ***routine générique*** qui, par l'intermédiaire de la routine spécifique dont le nom, `NOM_OPA`, est défini dans le fichier de données *cf.* §3.3 (Page 14), réalise la gestion du calcul de l'opacité. Pour un triplet: (composition chimique, température, densité), les routines de type `opa` déterminent l'opacité Rosseland $\kappa(T, \rho, X, Z)$, ainsi que ses dérivées premières par rapport à i) la température, ii) la densité et, iii) la teneur en hydrogène. Deux types d'opacités sont implantés dans Cesam2k20, des opacités analytiques et des opacités tabulées. Les premières sont des approximations, leur utilisation est robuste et ne demande que très peu de ressources; peu précises elles sont réservées aux tests. Les secondes, plus précises, mais d'utilisation délicate, sont destinées aux exploitations.

ORGANIGRAMME DE NUC

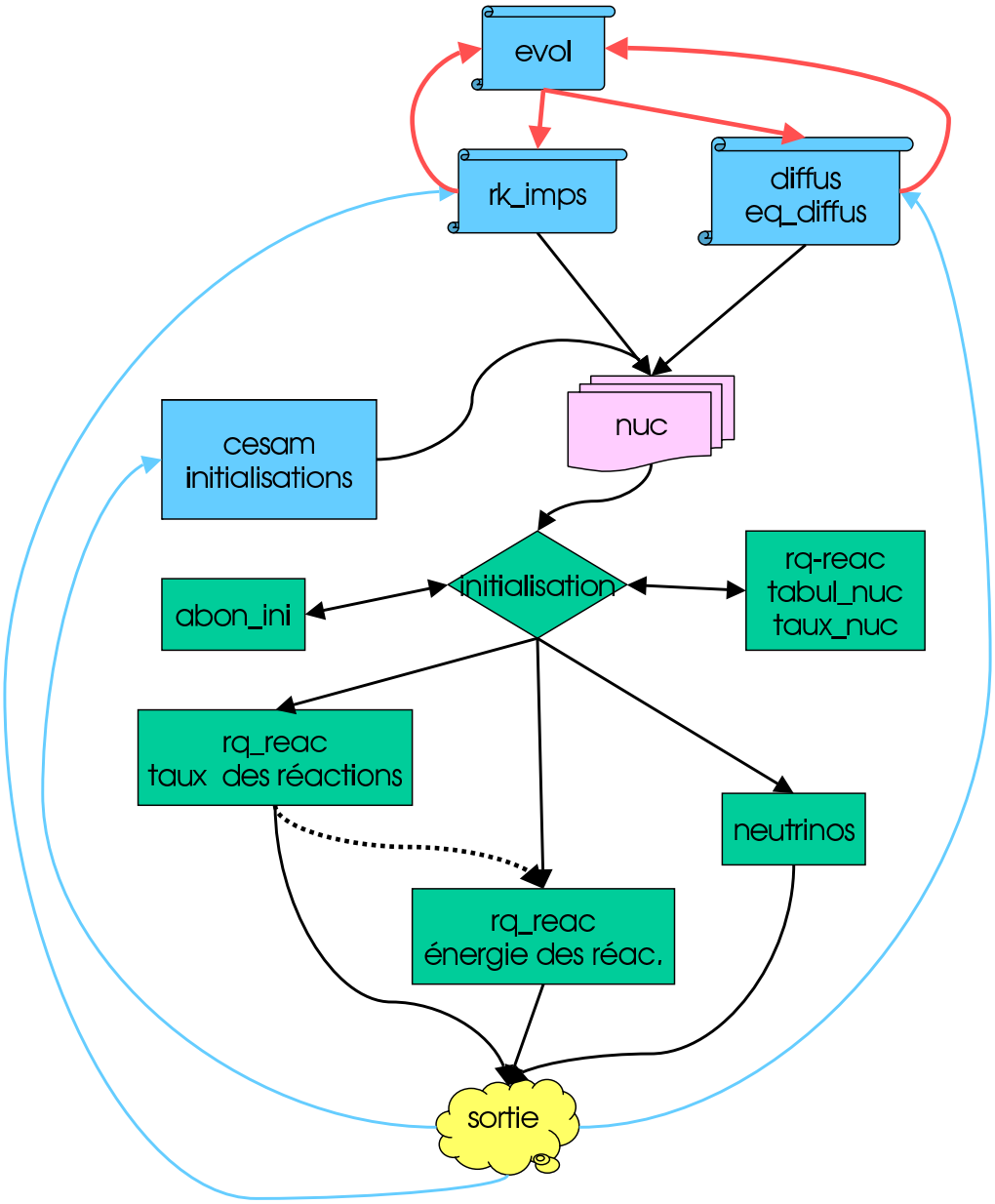


Figure 7.3: Environnement de la routine de calcul des réactions thermonucléaires. La routine `nuc` est appelée de `cesam` pour l'initialisation des abondances. Lors d'une évolution, elle est appelée i) de `eq_dif_chim` ou de `rk_imps` pour le calcul de l'évolution des abondances, suivant qu'il y ait ou non suivi de la diffusion des éléments chimiques, ii) de `thermo` pour le calcul de l'énergie thermonucléaire libérée, iii) de `list` pour le calcul des flux de neutrinos. Au premier appel, il y a initialisation des abondances initiales et tabulation des réactions nucléaires suivant le cycle et la compilation des réactions à utiliser, *cf.* § 3.3 (Page 14). Lors des appels suivants, il y a calcul des variations des abondances ou/et de l'énergie thermonucléaire libérée, ou encore des flux de neutrinos.

Description :
Les routines de type opa, la plupart d'origine externe, n'ont pas une structure commune; chacune d'entre elles fait l'objet d'une description personnalisée. L'opacité conductive, *cf.* §7.41 (Page 203), est introduite à l'issue du calcul de l'opacité radiative.

Appel :
opa est appelée de divers endroits en particulier de evol, thermo, thermo_atm, eq_dif_chim, cesam.

```
SUBROUTINE opa(xh,t,ro,kappa,dkapdt,dkapdr,dkapdx)
```

- Entrées :
  - xh, t, ro: composition chimique, température, densité.
- Sorties :
  - kappa,dkapdt,dkapdr,dkapdx: opacité, dérivées.

7.50 Routine opa_gong

La routine PRIVATE opa_gong du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité analytique. Le formalisme utilisé est décrit dans le "Solar Model Comparison Project" de Christensen-Dalsgaard (1988). L'opacité moyenne de Rosseland est approchée par une formule de Kramers améliorée, sous la forme d'une somme de deux termes, l'un valable pour l'intérieur stellaire et l'autre pour l'extérieur :

$$\kappa_{\rm R}^{-1} = \kappa_e^{-1} + \kappa_i^{-1}, \quad \kappa_e = c_e \rho^{m_e} T^{n_e}, \quad \kappa_i = c_i \rho^{m_i} T^{n_i},$$

avec: $c_e = 1.6236784 \times 10^{-33}$, $m_e = 0.407895$, $n_e = 9.28289$, $c_i = 7.1548412 \times 10^{13}$, $m_e = 0.138316$, $n_e = -1.97541$. Cette forme analytique correspond à $Z = 0.02$.

Appel :
opa_gong, appelée par la routine générique opa *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.51 Routine opa_houdek9

La routine PRIVATE opa_houdek9 du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité tabulée. Elle utilise les interpolations des tables de Livermore 95 prolongées, pour les basses températures, par les opacités Kurucz ou Alexander; elles peuvent, ou non, tenir compte de la conductivité électronique. Elles utilisent les bibliothèques et les données du package de **?**. La mise en forme des données est décrite au §**??** (Page **??**). Pour son utilisation, il est nécessaire d'indiquer dans la routine opa_houdek le chemin qui permet d'atteindre les données OPINPATH_95; il faut aussi indiquer au LINK le chemin de la bibliothèque libopint.a. La mise en oeuvre de opa_houdek a bénéficié d'une collaboration avec N.Audard. La version originale du package de routines d'interpolation de G.Houdek se trouve à: ftp://solaris.tuwien.ac.at/incoming/.

Description :

- Au premier appel :
  - Lecture et initialisation des tables.
- Adaptation et encadrement des données.

- Calcul de l'opacité, en cas d'échec appel à opa_yveline_lisse, *cf.* §7.56 (Page 213).

Les routines d'interpolation d'origine externe ne sont pas décrites. Pour plus d'informations, se référer aux README du packetage "v9" du sous-directory SUN_STAR_DATA.

Appel :

opa_houdek9, appelée par la routine générique opa *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.52 Routine opa_int_zsx

La routine PRIVATE opa__int_zsx du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité tabulée. opa_int_zsx interpole les ***anciennes*** tables Livermore 1991, prolongées par les opacités Kurucz pour les basses températures. Elle utilise une interpolation spline 4D linéaire[3] suivant les axes $(T_6, \ln \rho/T_6^3, X, Z)$, $T_6 \equiv T \times 10^{-6}$. Les données Livermore 1991 sont actuellement caduques. Des difficultés de convergence, observées principalement dans la restitution de l'atmosphère, peuvent provenir de l'interpolation linéaire - non dérivable aux points de la table - dans ces tables aux larges intervalles tabulaires.

Description :

- Au premier appel :
  - Initialisation de constantes et extension des tables par interpolation.
  - Mise en forme des tables pour l'interpolation.
  - Estimations des limites des tables.
- Localisation du point de calcul.
- Interpolation et calcul des dérivées.

Appel :

opa_int_zsx, appelée par la routine générique opa *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.53 Routine opa_opal2

La routine PRIVATE opa_opal2 du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité tabulée. Elle permet de dépasser le seuil $Z < 0.1$ des données de OPAL. Elle tient compte des modifications de composition chimique dues à la nucléosynthèse $H \rightarrow He \rightarrow C \rightarrow N \rightarrow O$. Elle ne permet le calcul de l'opacité que pour des températures supérieures à 1eV, le raccord avec des opacités basse température n'existant pas. Elles n'est donc pas utilisable pour les modèles comportant une atmosphère. Le calcul de l'opacité est basé sur les tables d'opacité de type 2 de Livermore, il utilise les routines du paketage z14xcotrin21, *cf.* §7.85 (Page 233), de A.I. Boothroyd. Les tables utilisées, placées dans le sous-directory SUN_STAR_DATA sont automatiquement décompressées et transformées en binaire à l'occasion de leur première utilisation.

opa_opal2 ***est d'usage extrèmement délicat.***

Description :

- Au premier appel :
  - Identification du chemin directory où se trouvent les fichiers de données.

[3] La base utilisée est duale, les calculs d'interpolation sont alors immédiats.

  - Identification des indices de C, N, O, estimation de l'abondance en masse des éléments de la mixture de GN93 qui ne sont pas pris en compte et du rapport [O/Fe].
  - Détermination des rapports C/Z, O/Z, pour les excès de C et O de la mixture utilisée par rapport à GN93.
  - Lecture en binaire des tables d'opacité concernées si elles existent, sinon création.
  - Calcul de X, Y, Z initiaux et définition des limites pour les sorties de table.
- Calcul de l'opacité.
- Estimation des dérivées par dérivation numérique.

Les routines d'interpolation, d'origine externe ne sont pas décrites, *cf.* §7.85 (Page 233).

Appel :

La liste d'appel de cette routine est celle de sa routine générique opa *cf.* §7.49 (Page 208), avec un argument supplémentaire, la variable logique cno qui permet une alternative :

- cno=.FALSE.: seules les variations de l'abondance en carbone sont prises en compte.
- cno=.TRUE.: les variations de l'abondance en carbone et en oxygène sont prises en compte.

En raison de l'absence d'un raccord avec des opacités basse température, opa_opal2 est utilisée pour prolonger les routines d'opacité opa_yveline, *cf.* §7.55 (Page 213), et opa_yveline_lisse, *cf.* §7.56 (Page 213), au delà de $Z > 0.1$.

Il est toutefois possible d'utiliser opa_opal2 directement, en codant dans le fichier de données, *cf.* §3.3 (Page 14), NOM_OPA=opa_opal2_cno, pour tenir compte seulement des variations de l'abondance du carbone, ou NOM_OPA=opa_opal2_co, pour tenir compte des variations de l'abondance du carbone et de l'oxygène.

7.54 Routine opa_opalCO

La routine PRIVATE opa_opalCO du module mod_opa, *cf.* §D.6 (Page 364), est du type opacité tabulée. Elle permet de dépasser le seuil $Z < 0.1$ des données de OPAL. Elle tient compte des modifications de composition chimique dues à la nucléosynthèse $H \rightarrow He \rightarrow C \rightarrow O$. Elle ne permet le calcul de l'opacité que pour des températures supérieures à leV, le raccord avec des opacités basse température n'existant pas. Elle est inutilisable pour les modèles comportant une atmosphère.

Les tables utilisées par ces routines, placées dans le sous-directory SUN_STAR_DATA sont automatiquement décompressées et transformées en binaire à l'occasion de leur première utilisation.

La mise en oeuvre de opa_opalCO a bénéficié d'une collaboration avec L.Piau.

Description :

- Au premier appel :
  - Identification des indices de H, He, C, O.
  - Définition de la métallicité de la table et calcul des rapports C/Z, O/Z.
- Détermination des abondances de H, He, C, O, et des excès de C et O.
- Calcul des opacités.

opa_opalCO utilise une interpolation moins sophistiquée des tables OPAL que opa_opal2. Les routines d'interpolation utilisées sont des routines subordonnées de opa_opalCO; d'origine externe, ces routines ne sont pas décrites. Il s'agit de :

opac, cointerp, t6rinterp, readco, quad, spline, splint, fity, fitx, getd, interp, smooth, opaltab.

Appel :

opa_opalCO, appelée par la routine générique opa *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.55 Routine opa_yveline

La routine PRIVATE opa_yveline du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité tabulée. L'interpolation du package construit par Y. Lebreton pour le calcul de l'opacité OPAL (**?**). opa_yveline effectue une interpolation cubique dans la table, linéaire aux bords.La table en ASCII se trouve dans le fichier opa_yveline_etendu.dat du sous-directory SUN_STAR_DATA. Extraire cette table et la transformer en un fichier binaire à l'aide du programme ascii2bin_opa de ce package *cf.* §**??** (Page **??**).

Lorsque l'abondance en éléments lourds dépasse $Z > 0.1$ opa_yveline fait appel à opa_opal2, *cf.* §7.53 (Page 211), qui permet de tenir compte de la nucléosynthèse $H \rightarrow He \rightarrow C \rightarrow N \rightarrow O$.

Description :

- Au premier appel :
  - Vérification de la cohérence des données.
- Vérification de l'accessibilité du point d'interpolation.
- Calcul de l'opacité si $Z < 0.1$.
- Appel à opa_opal2 si $Z > 0.1$.

Les routines d'interpolation utilisées par opa_yveline sont introduites sous forme de routines subordonnées, d'origine externe, elles ne sont pas décrites. Il s'agit de:
kappa_opal, intlin_opal, intl_opal, sub_quad, lect_opal, lpol_op, pos_table_op.

Appel :

opa_yveline, appelée par la routine générique opa *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.56 Routine opa_yveline_lisse

La routine PRIVATE opa_yveline_lisse du module mod_opa, *cf.* §D.6 (Page 364), est une routine de type opacité tabulée. L'exploitation du package construit par Y. Lebreton pour le calcul de l'opacité OPAL (**?**) et effectué par opa_yveline_lisse est un lissage par des Béziers, dont l'ordre peut être facilement adapté pour chacune des dimensions (X, Z, T, ρ). Par défaut l'ordre 4 est utilisé, l'ordre 2 correspond à l'interpolation linéaire. Le lissage assure des dérivées d'ordre élevé, il est plus robuste que l'interpolation. Le prix à payer est une précision réduite. La table en ASCII se trouve dans le fichier opa_yveline_etendu.dat du sous-directory SUN_STAR_DATA. Extraire cette table et la transformer en un fichier binaire à l'aide du programme ascii2bin_opa de ce package *cf.* §**??** (Page **??**).

Description :

- Au premier appel :
  - Lecture des tables.
  - Définitions diverses.

- Vérification de l'accessibilité du point d'interpolation.
- Appel à `opa_opal2` si $Z > 0.1$.
- Si $Z < 0.1$, recherche des indices, calcul de l'opacité.

Appel :
`opa_opa_yveline_lisse`, appelée par la routine générique `opa` *cf.* §7.49 (Page 208), utilise les mêmes arguments.

7.57 Routine `osc_adia, osc_invers, osc_noad`

Ces routines PRIVATE du module `mod_cesam`, *cf.* §D.13 (Page 368), de type `output`, *cf.* §7.58 (Page 214), ont pour fonction la création des fichiers utilisés respectivement pour le calcul des oscillations adiabatiques, des inversions et des oscillations non adiabatiques.

Description :
Les tableaux d'indices `tglob(nglob)` et `tvar(nvar)`, adaptés à chaque fichier de sortie, définissent l'ordre dans lequel sont disposés les éléments de ces fichiers ASCII. Les éléments des tableaux `var(P, T, L...)` et `glob(age, d2ro...)` construits dans la routine `cesam`, *cf.* §7.7.3 (Page 158), et complétés dans la routine `add_ascii`, *cf.* §7.7.1 (Page 157), sont réorganisés dans les tableaux intermédiaires `eglob(nglob)` et `evar(nvar,itot)`, suivant l'ordre défini dans `tglob(nglob)` et `tvar(nvar)`. Le fichier de sortie est ensuite créé à partir des fichiers intermédiaires. L'identification des quantités est décrite au §C.1 (Page 345). Un exemple de création de tels fichiers est décrit au §**??** (Page **??**).

Appel :
`osc_adia, osc_invers, osc_noad`, appelées par la routine générique `output` *cf.* §7.58 (Page 214), utilisent les mêmes arguments.

7.58 Routine générique `output`

Cette routine PRIVATE du module `mod_cesam`, *cf.* §D.13 (Page 368), est la ***routine générique*** des routines de formation des fichiers ASCII de sortie, utilisés pour les calcul des oscillations ou des inversions, *cf.* §7.57 (Page 214), ou encore pour créer un fichier de sortie ASCII personnalisé, *cf.* §**??** (Page **??**).

Description :
Suivant le type de sortie ASCII indiqué par la variable `NOM_OUTPUT` du fichier de données `mon_modele.don`, *cf.* §3.3 (Page 14), la routine de formation du fichier ASCII est appelée.

Appel :
`output` est appelée par la routine `ecrit_ascii`, *cf.* §7.19 (Page 172) subordonnée de la routine `cesam`; `output` n'a que des arguments d'entrée.

```
1 SUBROUTINE output(var,glob,itot,ivar)
```

- Entrées :
  - `glob, itot, ivar`: quantités globales, nombre total de points, nombre de variables

7.59 Routine générique `pertm`

Cette routine PUBLIC du module `mod_static`, *cf.* §D.12 (Page 367), est la ***routine générique*** du calcul de la perte de masse.

Description :
Suivant le type de sortie ASCII indiqué par la variable NOM_PERTM du fichier de données mon_modele.don, *cf.* §3.3 (Page 14), la routine de perte de masse est appelée.
Appel :
pertem est appelée par evol, *cf.* §7.31.3 (Page 183), et resout, *cf.* §7.67 (Page 219).

```
SUBROUTINE pertem(dt))
```

- Entrée
  - dt: pas temporel

7.60 Routine pertm_ext, pertm_msol

Pour ces deux routines PRIVATE du module mod_static, *cf.* §D.12 (Page 367), la perte de masse est supposée linéaire en fonction du temps. Le taux mdot$M_\odot$ par an est celui indiqué dans le fichier de données mon_modele.don, *cf.* §3.3 (Page 14). Avec pertm_msol, le taux de perte de masse ainsi que le traitement du vent *cf.* §3.15.4 (Page 32), sont annulés dès que la masse du modèle atteint la masse solaire.
Description :

- Extraction des masses.
- Formation du tableau des masses corrigées de la perte/gain de masse.

Appel :
Ces deux routines ont les mêmes arguments que leur routine générique pertm, *cf.* §7.59 (Page 214).

7.61 Routine pertm_tot

Cette routine PRIVATE du module mod_static, *cf.* §D.12 (Page 367), tient compte de la masse perdue par la transformation de la masse en énergie, par les réactions thermonucléaires. La perte de masse externe est supposée linéaire en fonction du temps. Le taux mdot$M_\odot$ par an est celui indiqué dans le fichier de données mon_modele.don, *cf.* §3.3 (Page 14).
Description :

- Extraction des masses.
- Calcul de la perte de masse locale due aux réactions nucléaires.
- Formation du tableau des masses corrigées de la perte/gain de masse totale.

Appel :
Cette routine a les mêmes arguments que sa routine générique pertm, *cf.* §7.59 (Page 214).

7.62 Routine pertem_waldron

Cette routine PRIVATE du module mod_static, *cf.* §D.12 (Page 367), utilise la perte de masse empirique de Waldron, citée dans l'article A.Ap 229 (1990) 469-474.
Description :

- Extraction des masses.
- Calcul de la perte de masse.
- Formation du tableau des masses corrigées de la perte de masse.

Appel :
Cette routine a les mêmes arguments que sa routine générique `pertm`, *cf.* §7.59 (Page 214).

7.63 Routine `planetoides`

Cesam2k20 offre la possibilité de tenir compte des variations de composition chimique dues à des chutes de planétoïdes dans la zone convective externe, *cf.* §6.7.2 (Page 87). Cette possibilité est gérée par la routine PUBLIC `planétoides` du module `mod_nuc`, *cf.* §D.9 (Page 365).
Description :

- Lors d'un appel d'initialisation, depuis la routine `evol`, on effectue une recherche dans l'environnement du fichier où sont indiquées les fractions de masse des planétoïdes, *cf.* §3.15.5 (Page 33). Si un tel fichier est trouvé, les fractions de masse sont normalisées, puis réparties entre les différents isotopes du réseau nucléaire, et enfin transformées en fraction par mole; la variable logique `l_planet` est déclarée `.TRUE.`
- Lors des appels suivants, une correction aux taux de réactions nucléaires est effectuée lorsque l'âge du modèle est dans l'intervalle de temps des chutes. *cf.* §6.7.2 (Page 87).

Appel :
`planetoides` est appelée par :

- `evol`, *cf.* §7.31.3 (Page 183), pour initialisation.
- `rk_imps`, *cf.* §7.71 (Page 222) ou `eq_dif_chim`, *cf.* §7.17 (Page 171), à la suite d'un appel à la routine `nuc`, *cf.* §7.6.1 (Page 149).

Les arguments sont optionnels.

```
1 SUBROUTINE planetoides(xchim,dxchim,jac,m_planet,mw_planet)
```

- Entrées :
  - `xchim`: Composition chimique par mole.
- Entrées/Sorties :
  - `dxchim, jac`: Taux de réactions nucléaires et jacobien.
  - `m_planet,mw_planet`: taux d'accroissement de masse et de moment cinétique.

7.64 Routine `poisson_initial`

Cette routine PRIVATE du module `mod_cesam`, permet l'initialisation du potentiel gravitationnel Ψ dans le cadre du formalisme de Mathis & Zahn (2004) de la diffusion du moment cinétique. L'intégration de léquation différentielle utilise la méthode des éléments finis Galerkin.
Description :

- initialisations.

- formation des produits scalaires avec appel à eq_diff_poisson.f90, *cf.* §7.22 (Page 174).
- résolution du sytème linéaire.
- test de précision et gestion des itérations.
- initialisation des coefficients du potentiel gravitationnel.

Appel :
poisson_initial est appelée de la routine cesam.f90, *cf.* §7.7.3 (Page 158), lors des initialisations.

7.65 Routine print_ctes

Cette routine PUBLIC du module mod_donnees, *cf.* §D.3 (Page 353), a pour fonction la transcription sur l'unité d'écriture des valeurs des principales constantes physiques utilisées.

Appel :
print_ctes est appelée de cesam, *cf.* §7.7.3 (Page 158)

```
SUBROUTINE print_ctes(i)
```

- Entrée
  - i: indice de l'unité FORTRAN sur laquelle l'écriture est effectuée.

7.66 Routine read_ascii

Cette routine PUBLIC du module mod_exploit, *cf.* §D.14 (Page 368), a pour fonction la lecture des fichiers ASCII d'oscillation, *cf.* §7.57 (Page 214).

Description :

- Ouverture du fichier de type mon_modele.osc, lecture de l'en-tête et des constantes.
- Lecture des variables.

Appel :
read_osc est appelée par divers programmes de dessin, en particulier des_osc, des_abon, des_diff_osc ou encore lit_osc.

```
SUBROUTINE read_ascii(nom_fich,itot,nglob,nvar,abid)
```

- Entrée
  - nom_fich: nom du fichier de type mon_modele.osc (sans l'extension .osc).
- Sorties :
  - itot, nglob, nvar, abid: nombre de points, de "constantes global", de variables du tableau var, sans les éléments chimiques,

La description des quantités extraites de ces fichiers est donnée au §C.1 (Page 345).

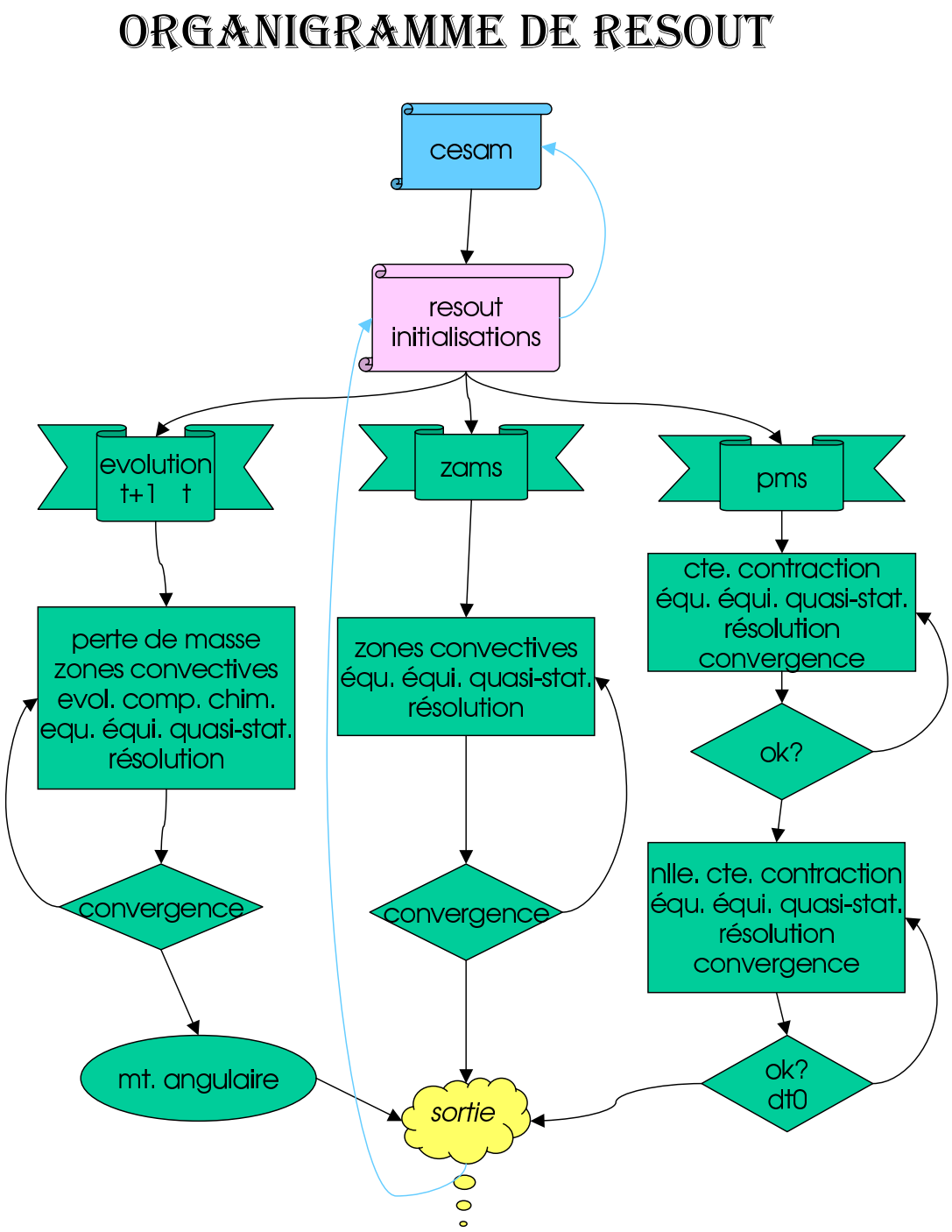


Figure 7.4: Organigramme de la routine de la gestion de la résolution des équations de la structure interne `resout`.

7.67 Routine resout

La fonction de la routine PRIVATE resout du module mod_static, *cf.* §D.12 (Page 367), est la gestion de la résolution des équations de l'évolution stellaire pour :

1. Le modèle en cours d'évolution.
2. Le modèle de séquence principale d'âge zéro homogène.
3. Le modèle initial de PMS homogène.

La figure 7.4 (Page 218), présente un organigramme de la routine resout.

Description :

L'architecture des algorithmes de résolution des équations de la structure interne est assez similaire dans les trois phases de l'évolution évoquées çi-avant. La structure de chacune d'entre elles est celle d'une itération Newton-Raphson.

- Initialisation de divers paramètres généraux.
- Poursuite d'une évolution :
  - Ajustement de paramètres s'il s'agit du calcul du modèle ultime.
  - Appel à update, *cf.* §7.83 (Page 232) pour mise en mémoire du modèle au pas temporel précédent.
  - Initialisation de compteurs et début de l'itération Newton-Raphson.
  - Calcul de la perte de masse.
  - Estimation du nombre de couches nécessaire pour assurer à la constante de répartition la valeur définie par la précision requise, *cf.* §6.2.4 (Page 62).
  - Appel à lim_zc, *cf.* §7.42 (Page 204), pour un réaménagement éventuel de la discrétisation spatiale, la détection des limites zones radiatives / zones convectives et l'ajustement des facteurs de répartition, *cf.* §6.2.5 (Page 65).
  - Evolution de la composition chimique, éventuellement du moment cinétique, par appel à evol, *cf.* §7.31.3 (Page 183).
  - Formation et résolution des équations de l'équilibre quasi-statique par appel à coll_qs, *cf.* §7.33.1 (Page 199).
  - Suivant l'estimation de la précision :
    - Ajustement du pas temporel pour satisfaire les conditions d'arrêt particulières, réinitialisation et reprise des itérations Newton-Raphson.
    - Poursuite des itérations Newton-Raphson.
    - Réinitialisation avec diminution du pas temporel et reprise.
    - Gestion éventuelle de la rotation solide, estimation de l'énergie graviphique, tabulation de diverses quantités, retour vers cesam, *cf.* §7.7.3 (Page 158).
    - Appel à sortie, *cf.* §7.74 (Page 225) pour arrêt du calcul.
- Modèle initial de séquence principale d'âge zéro homogène, ou de PMS homogène :
  - Estimation du nombre de couches nécessaires pour assurer à la constante de répartition la valeur définie par la précision requise, *cf.* §6.2.4 (Page 62).
  - Appel à lim_zc, *cf.* §7.42 (Page 204), pour un réaménagement éventuel de la discrétisation spatiale, la détection des limites zones radiatives / zones convectives et l'ajustement des facteurs de répartition, *cf.* §6.2.5 (Page 65).

- Initialisation de compteurs et début de l'itération Newton-Raphson.
- Formation et résolution des équations de l'équilibre quasi-statique par appel à `coll_qs`, *cf.* §7.33.1 (Page 199).
- Suivant l'estimation de la précision :
  * Poursuite des itérations Newton-Raphson, précédée par une réactualisation de la localisation des limites zones radiatives / zones convectives par appel à `lim_zc`.
  * Gestion éventuelle de la rotation solide, estimation de l'énergie graviphique, tabulation de diverses quantités, retour vers `cesam`, *cf.* §7.7.3 (Page 158).
  * Appel à `sortie`, *cf.* §7.74 (Page 225), pour arrêt du calcul.

Appel :
À diverses reprises `resout` est appelée par `cesam`, *cf.* §7.7.3 (Page 158).

```
1 SUBROUTINE resout(un23,dt,dts)
```

- Entrées :
  - `un23,dt,`: Paramètre du type de modèle, pas temporel.
- Sorties :
  - `dts`: Estimation de la valeur du pas temporel suivant.

7.68 Routine `resout_chim`

Cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), a pour fonction la résolution par éléments finis du système implicite des équations de la diffusion des espèces chimiques, *cf.* §6.7 (Page 84). Les produits scalaires sont approchés par une intégration numérique de type Gauss, suivant un algorithme inspiré de l'algorithme 5.22 p.203 de Schumaker Schumaker (alg. 5.22 1981, p.203). De structure identique à `resout_rota`, *cf.* §7.69 (Page 221), ces deux routines pourraient être rassemblées en une seule, elles sont distinctes pour faciliter la lisibilité.

Description :
L'architecture globale de la routine est celle d'une itération Newton-Raphson. Les coefficients provisoires des B-splines, *i.e.* les inconnues, sont initialisés dans la routine d'appel `diffus`, *cf.* §7.17 (Page 171).

- Initialisations au premier appel.
- Définition du rang du système, du nombre de lignes, des indices de première colonne, initialisation de compteurs.
- Itérations Newton-Raphson :
  - Pour chaque intervalle du vecteur nodal, détermination des abscisses de Gauss, formation des coefficients des équations par appel à `eq_diff_chim`, *cf.* §7.21 (Page 173).
  - Pour chaque B-spline non identiquement nulle sur l'intervalle, évaluation du second membre du système linéaire et de sa contribution au jacobien.
  - Résolution du système linéaire bande, correction des coefficients provisoires des B-splines, Estimation de la précision.

- Suivant la précision obtenue, poursuite des itérations Newton-Raphson, retour vers la routine appelante, réinitialisation si la précision requise n'a pas été atteinte après un nombre fixé d'itérations.

Appel :
`resout_chim` est appelée par `diffus`, *cf.* §7.17 (Page 171).

```
SUBROUTINE resout_chim(dt,ok)
```

- Entrées :
  - `dt`: Pas temporel.
- Sorties :
  - `ok=.TRUE.`: La convergence a été obtenue.

7.69 Routine `resout_rota`

Cette routine générique PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), a pour fonction le choix entre les deux formalismes de diffusion du moment cinétique implanté dans Cesam2k20.

Description :
Suivant la valeur de la variable `Krot`, il y a choix entre le formalisme de Talon et al. (1997) ou celui de Mathis & Zahn (2004).

Appel :
`resout_rota` est appelée par `diffus`, *cf.* §7.17 (Page 171).

```
SUBROUTINE resout_rota(dt,ok)
```

- Entrées :
  - `dt`: Pas temporel.
- Sorties :
  - `ok=.TRUE.`: La convergence a été obtenue.

7.70 Routine `resout_rota3/4`

Ces routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), ont pour fonctions la résolution par éléments finis du système implicite des équations de la diffusion du moment cinétique suivant les formulations de **??**, *cf.* §6.8 (Page 98). Les produits scalaires sont approchés par une intégration numérique de type Gauss, suivant un algorithme inspiré de Schumaker (Alg. 5.22 1981, p. 203). De structure identique à `resout_chim`, *cf.* §7.68 (Page 220), ces routines pourraient être rassemblées en une seule, elles sont distinctes pour faciliter leur lisibilité.

Description :
L'architecture globale de la routine est celle d'une itération Newton-Raphson. Les coefficients provisoires des B-splines, *i.e.* les inconnues, sont initialisés dans la routine d'appel `diffus`, *cf.* §7.17 (Page 171).

- Initialisations au premier appel.

- Définition du rang du système, du nombre de lignes, des indices de première colonne, initialisation de compteurs.
- Itérations Newton-Raphson :
  - Pour chaque intervalle du vecteur nodal, détermination des abscisses de Gauss, formation des coefficients des équations par appel à `eq_diff_rota`, *cf.* §7.23 (Page 175).
  - Pour chaque B-spline non identiquement nulle sur l'intervalle, évaluation du second membre du système linéaire et de sa contribution au jacobien.
  - Pour le centre, formation des coefficients des parties intégrées par appel à `eq_diff_rota3/4`, *cf.* §7.23 (Page 175), et calcul des contributions au second membre et au jacobien du système linéaire.
  - Pour la surface, formation des coefficients des parties intégrées par appel à `eq_diff_rota3/4`, *cf.* §7.23 (Page 175), et calcul des contributions au second membre et au jacobien du système linéaire.
  - Résolution du système linéaire bande, correction des coefficients provisoires des B-splines, estimation de la précision.
- Suivant la précision obtenue, poursuite des itérations Newton-Raphson, retour vers la routine appelante, réinitialisation si la précision requise n'a pas été atteinte après un nombre fixé d'itérations.

Appel :
`resout_rota3/4` sont appelées par `resout_rota` et ont les mêmes arguments.

7.71 Routine `rk_imps`

Cette routine PRIVATE du module `mod_evol`, *cf.* §D.11 (Page 367), a pour fonction l'intégration des équations de l'évolution des espèces chimiques *e.g.* Eq. 6.294 (Page 128) en l'absence de diffusion. Les échelles de temps d'évolution des divers isotopes diffèrent par plusieurs dizaines de magnitudes. A titre d'exemple, dans le soleil, l'échelle de temps d'évolution de ^{2}H est de quelques minutes, alors que celle de ^{1}H est de l'ordre de la centaine de million d'années. La disparité de ces échelles de temps est à l'origine de difficultés numériques: si on s'impose de suivre la plus courte échelle de temps, le temps de calcul est inimaginable, si on s'impose de suivre la plus grande échelle de temps, les erreurs numériques engendrées sur les espèces chimiques avec courte échelle de temps font "exploser" le calcul.

Pour ce type de problème numérique, qualifié de "raide", ou encore de "mal posé", il existe des algorithmes spécifiques qui, *en gros*, permettent de suivre l'échelle de temps que l'on désire, *e.g.* celle de ^{1}H pour le soleil, sans que, pour autant, les erreurs numériques engendrées sur les espèces chimiques à courte échelle de temps ne perturbent la solution de façon catastrophique. Ces algorithmes sont implicites, ils nécessitent le calcul d'un Jacobien; en cas de non linéarité, la solution est obtenue par approximations successives, le plus souvent par la méthode de Newton-Raphson, qu'il convient parfois d'aménager.

Dans le cas de la structure interne, le mélange convectif complique encore le problème qui, non seulement est raide, mais est donc aussi intégral. En résumé, le problème du calcul de l'évolution des espèces chimiques est un problème intégro-différentiel aux valeurs initiales, non linéaire, raide.

La routine `rk_imps` utilise le formalisme de Runge-Kutta implicite avec le schéma Lobatto IIIC, *cf.* Hairer & Wanner (1996), ainsi qu'il a été décrit au §6.5.3 (Page 76), noté dans la suite IRK Lobatto IIIC. L'ordre d'intégration est limité au troisième ordre. L'utilisation d'un ordre supérieur nécessiterait l'interpolation de la température, de la densité et de la composition chimique en tous les points du

modèle entre différents pas temporel, interpolations dans lesquelles serait perdu le gain en précision que l'on peut attendre de l'utilisation d'un schéma d'ordre élevé. Cette routine originale est l'une des plus "techniques" et élégantes de Cesam2k20.

Choix du pas temporel :

La valeur du pas temporel à utiliser dépend de la précision requise. A l'issue de chaque intégration, la précision obtenue est estimée. Dans le programme appelant `evol`, *cf.* §7.31.3 (Page 183), le pas temporel est alors diminué (*respt.* augmenté) si la précision obtenue est inférieure (*respt.* supérieure) à ce qui est souhaité. La précision d'une intégration peut être estimée en l'effectuant avec des algorithmes différents, ou des pas temporels différents, ce qui nécessite de refaire plusieurs fois le calcul. Pour limiter l'effort de calcul, avec `rk_imps` on se contente de contrôler le pas temporel en limitant les variations de certains isotopes au cours d'un pas temporel. Cette méthode, quoique très répandue, n'est pas correcte car une grande (*respt.* petite) variation d'une abondance ne signifie pas, ***nécessairement***, que l'intégration est imprécise (*respt.* précise).

Description :

- Initialisations des matrices de IRK Lobatto IIIC suivant l'ordre de précision, `ordre`, défini dans la routine `cesam`, *cf.* §7.7.3 (Page 158), des contraintes du schéma de Newton-Raphson et de l'ordre de grandeur de chaque abondance utilisée pour estimer la précision relative de l'intégration et de la convergence.
- Pour chaque temps intermédiaire, appel à la routine de réactions thermonucléaires pour le calcul des Ψ_i, éventuellement appel à la routine `vent`, *cf.* §3.15.4 (Page 32), pour tenir compte de la perte ou du gain de masse, calcul de la contribution à la ligne correspondante du jacobien, détermination de la contribution à l'intégrale de Gauss dans les zones convectives.
- Formation de la ligne correspondante de Eq. 6.78 (Page 77).
- Solution du sytème linéaire donnant les corrections du processus itératif.
- Estimation de la précision et des nouvelles abondances, et poursuite, si besoin, du processus itératif.

Arguments de `rk_imps`

```
SUBROUTINE rk_imps(t_t,ro_t,compx,t,ro,compy,dt,esti,ok,nuc,kk,z_vent,dm)
```

- Entrées :
  - `t_t, ro_t, t, ro`: tables des températures, densité aux temps t et $t+dt$;
  - `dt, kk, dm, compx, z_vent`: pas temporel, nombre de couches à mélanger, tables des intervalles de masse et de la composition chimique au temps t, variable logique indiquant que l'on doit corriger les taux de réactions nucléaires, pour tenir compte d'un vent de composition chimique différente de celle des couches extrernes *cf.* §?? (Page ??).
- Entrées/Sorties :
  - `compy`: table des la composition chimique au temps $t+dt$.
- Sorties :
  - `ok=.TRUE.`: la précision requise est atteinte.

Appel :
`rk_imps` est appelée par `evol`.

7.72 Routine rq_reac

La fonction de cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est la gestion du calcul, en fonction de la température, de la densité et des abondances, des taux des réactions nucléaires utilisées et des énergies libérées, ainsi que de leurs dérivées par rapport à la température, à la densité et aux abondances. Suivant la valeur de la variable logique mitler=.FALSE. ou mitler=.TRUE. de la NAMELIST NLCHIM du fichier de données mon_modele.don, *cf.* §3.3 (Page 14), l'écrantage est calculé soit par la théorie standard de l'écrantage faible, soit par celle, plus universelle, de l'écrantage de Mitler (1977) qui recouvre l'écrantage faible et intermédiaire[4]. Suivant les réactions utilisées, *cf.* §7.6.1 (Page 149), la routine utilise les tabulations des taux de réactions nucléaires gérées par la routine tabul_nuc, *cf.* §7.77 (Page 227).

Description :

- Initialisation: lecture du fichier de réactions nucléaires, écritures et initialisations diverses.
- Calcul du nombre d'électrons libres, du facteur d'écran et des dérivées.
- Calcul des taux et des énergies libérées. C'est dans la tabulation des réactions nucléaires, *cf.* §7.79 (Page 228), que sont introduits les ***dénominateurs 2! ou 3!*** et les pertes par neutrinos.

Appel :

```
SUBROUTINE rq_reac(comp,t,ro,r,drt,dro,drx,q,dqt,dqo,dqx,mue,dmuex)
```

- Entrées :
  - comp, t, ro: Composition chimique par volume, température, densité.
- Sorties :
  - r, drt, dro, drx, q, dqt, dqo, dqx, mue, dmuex: Taux, énergie des réactions et dérivées par rapport à la température, densité, composition chimique, nombre d'électrons e^- par volume, et dérivées par rapport à la composition chimique.

7.73 Routine saha

Cette routine PUBLIC du module mod_etat, *cf.* §D.5 (Page 363), résout l'équation de Saha (Cox & Giuli, 1968, eq. 5.30) pour la détermination des taux d'ionisation des divers éléments chimiques. On limite les fonctions de partition aux poids statistiques des niveaux fondamentaux. La formulation est décrite au §6.7.4 (Page 88). Les équations, écrites sous la forme:

$$\rho N_0 \sum_i \bar{z}_i(n_e) x_i - n_e = 0 \tag{7.6}$$

sont résolues par itération Newton-Raphson.

Description :

- Initialisations au premier appel :
  - On admet que les taux d'ionisation sont identiques pour tous les isotopes d'une même espèce chimique. Les différents isotopes d'une même espèce chimique sont regroupés dans un seul élément auquel l'abondance totale de l'espèce est affectée. Cesam2k20 signale si un des éléments utilisés est inconnu, le déroulement du calcul est alors arrêté.

[4]Bien que prévu, l'écrantage fort n'est pas opérant dans la version actuelle.

- Initialisation des potentiels d'ionisation et des poids statistiques pour les espèces identifiées par leur charge. La fonction de Fermi-Dirac $F_{1/2}$ est ensuite tabulée en fonction du paramètre de dégénérescence.

• Pour les appels suivants :

- Initialisation des abondances de chaque espèce et du nombre d'électrons libres.
- Calcul du facteur de dégénérescence et résolution des équations de Saha pour chaque ion, redétermination du nombre d'électrons libres. Itération Newton-Raphson sur le nombre d'électrons libres.
- Détermination de la charge moyenne de chaque ion.
- Après convergence: Restitution des taux d'ionisation et de la charge moyenne pour chaque isotope.

Appel :
saha est appelée par `cesam, coeff_rota, diffm_br`.

```
SUBROUTINE saha(xchim,t,ro,ioni,z_bar,nel,eta)
```

• Entrées :

- `xchim, t, ro`: composition chimique par volume, température, densité;

• Sorties :

- `ioni, z_bar, nel, eta`: taux d'ionisation et charge moyenne de chaque ion, nombre d'électrons libres, facteur de dégénérescence.

7.74 Routine `sortie`

La routine PUBLIC `sortie` du module `mod_variables`, *cf.* §D.4 (Page 360), est sans argument. Elle est appelée lorsque Cesam2k20 détecte une anomalie fatale au bon déroulement du calcul et à laquelle il ne peut remédier, par exemple une sortie de table. `sortie` ferme les unités d'écriture et de dessin avant l'arrêt des calculs.

7.75 Routine générique `static`

Cette routine PRIVATE du module `mod_static`, *cf.* §D.12 (Page 367), constitue la ***routine générique*** du calcul des coefficients des équations de l'équilibre quasi-statique dans l'espace physique.

Description :
`static` oriente le calcul des coefficients des équations d'équilibre quasi-statique vers leurs formes lagrangienne ou eulérienne, *cf.* §6.2.4 (Page 62).

Appel :

```
SUBROUTINE static(fait,cx,li,y,be,ae,compt,dt,reprend,ip)
```

• Entrées :

- `fait=1`: Calcul des résidus des équations au point de collocation.
- `fait=2`: Calcul du résidu pour la condition au point limite.

- cx: Indice du point de collocation.
- li: Numéro de la limite.
- y: variables et dérivées au point de collocation ou limite.
- compt: Compteur du nombre d'itérations Newton-Raphson globales.
- dt: Pas temporel.
- ip: Indice du facteur de répartition.

• Sorties :

- be: Résidus.
- ae: Eléments du jacobien.
- reprend=.TRUE.: La variation relative d'énergie graviphique est trop forte, le calcul devra être réinitialisé.

7.76 Routines static_m, static_r

Ces deux routines PRIVATE du module mod_static, *cf.* §D.12 (Page 367), sont équivalentes, elles forment les coefficients des équations de l'équilibre quasi-statique, sous leur forme lagrangienne et eulérienne respectivement, *cf.* §D.12 (Page 367). Pour chaque point de collocation, les équations de la structure interne sont écrites sous la forme $f(x) = 0$, ainsi que les conditions limites. On utilise les variables retenues pour l'intégration numérique. Ces équations ont été détaillées §6.2 (Page 57). Bien que, formellement, la variable indépendante soit la masse (lagrangien), dans static_m et le rayon (eulérien) dans static_r, les équations sont écrites pour "l'indice réel de couche" q, *cf.* §6.2 (Page 57).

Les singularités centrales des équations de la structure interne écrites sous la forme Eq. 6.14 (Page 58) disparaissent sous la forme Eq. 6.27 (Page 62) utilisée par Cesam2k20. Par ailleurs, le centre n'étant pas un point de collocation, il n'y a pas lieu d'introduire explicitement les formes limites des équations. Au centre, les variables ont pour valeurs limites celles, ***sans singularité***, des polynômes par morceaux qui approchent la solution, ce qui constitue un des avantages de la méthode.

Pour les limites externes, les fonctions qui dépendent de la physique utilisée: $P_{\rm b}(L,R)$, $P_{\rm gaz\ b}(L,R)$, $T_{\rm b}(L,R)$ et $M_{\rm b}(L,R)$ sont calculées, au point $q = n$ par une des routines de physique de type lim_atm, *cf.* §7.10.2 (Page 166).

Description :

Après quelques initialisations, on détermine la fonction de répartition et ses dérivées, puis on forme les équations et leurs dérivées par rapport à toutes les variables dépendantes. Les équations pour les conditions limites sont écrites à la fin de la routine. On a conservé dans la source des instructions permettant des vérifications, en particulier celles des coefficients du jacobien.

• Initialisations de quantités invariantes.

• Pour chaque point de collocation:

- Extraction des variables physiques et vérification que le rayon, la masse et la luminosité[5], ont des valeurs positives.
- Extraction de la composition chimique locale, initialisation de quantités liées à la rotation, à la pression turbulente.

[5]Pour static_m

- Calcul par un appel à thermo, *cf.* §7.80 (Page 229), des valeurs locales de grandeurs thermodynamiques, de l'opacité, de l'énergie nucléaire, des gradients et de leurs dérivées, transformations de ces grandeurs des unités cgs. en unités locales avec des aménagements algorithmiques.
- Calcul de l'énergie gravifique :
  * Pour un nouveau pas temporel, à la première itération globale gérée par resout, *cf.* §7.67 (Page 219), *i.e.* new=.TRUE., la valeur est interpolée, en masse, à partir de ses valeurs au pas temporel précédent.
  * Pour les itérations suivantes, cette quantité est calculée avec l'alternative de l'approximation de Kippenhahn *cf.* §6.4.1 (Page 72) ou du calcul complet *cf.* §6.4.2 (Page 73). Au cas où, à la même abscisse lagrangienne, un écart relatif trop important de pression ou de température est observé avec le pas temporel précédent cf, §5.3 (Page 43), la variable logique reprend=.TRUE. et les calculs sont réinitialisés par un retour à resout.
- Après quelques dispositions algorithmiques concernant le moment cinétique, détermination de la fonction de répartition et des dérivées, puis formulation des équations et de leurs dérivées partielles par rapport à chacune des variables.
- Ensemble d'instructions permettant de tester numériquement l'exactitude des dérivées.

• Pour chaque point limite:

- Ecriture des conditions limites au centre *i.e.* li=1, 2, 3.
- Ecriture des conditions limites à l'extérieur *i.e.* li=4, 5, 6. Pour limiter le nombre de calculs de l'atmosphère, lim_atm, *cf.* §7.10.2 (Page 166), n'est appelée qu'une seule fois, les grandeurs alors calculées étant utilisées pour formuler les autres conditions externes. Il suffit d'enlever les c de commentaires pour rendre effectifs des tests de vérification des dérivées.

Appel :
Ces routines appelées par la routine générique static, *cf.* §7.75 (Page 225), utilisent les mêmes arguments.

7.77 Routine tabul_nuc

La fonction de cette routine PRIVATE du module mod_nuc, *cf.* §D.9 (Page 365), est la gestion de la tabulation des réactions nucléaires.
Description :

• Pour chaque réseau de réactions nucléaires :

- Initialisation du nombre de réactions, du nombre d'isotopes utilisés par les réactions, des indices de ^{4}He et de la réaction 3α.
- Initialisation des indices des isotopes utilisés dans les réactions.
- Définition de l'ordre des réactions et du domaine de tabulation.

• Sélection du type de compilation à utiliser.

• Appel à taux_nuc, *cf.* §7.79 (Page 228), pour le calcul des taux de ***toutes*** les réactions implémentées.

• Extraction des taux des réactions relevant du réseau retenu et tabulation.

Appel :
cette routine d'initialisation `tabul_nuc` est appelée par `rq_reac`, *cf.* §7.72 (Page 224), en début de calcul. Cette routine n'a que des arguments de sortie.

```
SUBROUTINE tabul_nuc(ar,i3al,knot_temp,m_temp,
     1 nom_reac,n_temp,q0,taux_reac,temp,ttemp)
```

- `nom_reac, i3al`: Noms des réactions du réseau utilisé et indice de la réaction 3α.
- `ar, q0, taux_reac`: Tabulation des masses réduites, des énergies, des taux de réaction.
- `knot_temp, m_temp, n_temp, temp, ttemp`: Eléments du vecteur nodal de la tabulation des taux de réaction.

7.78 Routine `taueff`

Cette routine PRIVATE du module `mod_atm`, *cf.* §D.8 (Page 364), a pour fonction la détermination de la profondeur optique $\tau_\star$ correspondant au rayon bolométrique du modèle $R_\star$. $\tau_\star$ vérifie l'équation implicite :

$$T_{\rm eff} = \mathcal{T}(\tau_\star, T_{\rm eff}, g). \tag{7.7}$$

Description :

- Initialisation $\tau_\star = 2/3$.
- Détermination de $\tau_\star$ par l'algorithme de Newton-Raphson avec appel à la routine de la loi $T(\tau)$ concernée, *cf.* §7.10 (Page 162).

Appel :
`taueff` est appelée par `lim_atm`, *cf.* §7.10.2 (Page 166).

```
SUBROUTINE taueff(teff,grav,tau)
```

- Entrées :
  - `teff, grav`: température effective, gravité.
- Sorties :
  - `tau`: $\tau_\star$.

7.79 Routine `taux_nuc`

La fonction de cette routine PRIVATE du module `mod_nuc`, *cf.* §D.9 (Page 365), est l'initialisation des paramètres nucléaires, le calcul des énergies et des taux d'un grand nombre de réactions. Il est évidemment possible d'implémenter de nouvelles réactions, la procédure à suivre en est détaillée au §4.1 (Page 37).

Description :
Après une initialisation, les taux de réactions sont dérivés des formules de diverses compilations. C'est la nécessité du calcul des dérivées qui a motivé la tabulation des taux de réactions nucléaires plutôt que l'emploi direct des formules.

- Au premier appel :
  - Initialisations diverses.
  - Initialisation des excès de masse.
  - Initialisation des masses atomiques.
  - Initialisation des charges des isotopes.
  - Initialisation des noms des isotopes.
  - Initialisation des noms des réactions, des masses réduites, des énergies, des charges des noyaux de la réaction.
- Initialisations diverses reliées à la température.
- Calcul des taux de ***toutes*** les réactions, suivant la compilation retenue.

Appel :
`taux_nuc` est appelée par `tabul_nuc`, *cf.* §7.77 (Page 227).

```
SUBROUTINE taux_nuc(t,total,rt,zit,izzt,qt,nom_react,nucleot,
    1 at,nom_elemt)
```

- Entrées :
  - `t, total`: température, indice du choix de la compilation à utiliser, *cf.* §3.3 (Page 14).
- Sorties :
  - `rt`: ln(taux des réactions).
  - `zit`: Charges des noyaux.
  - `izzt`: Charges des noyaux de la réaction.
  - `qt`: Energies.
  - `nom_react`: Symboles des réactions.
  - `nucleot`: Masses des isotopes .
  - `at`: Masses réduites.
  - `nom_elemt`: Noms des isotopes.

7.80 Routine `thermo`

Cette routine PRIVATE du module `mod_static`, *cf.* §D.12 (Page 367), effectue le calcul des variables thermodynamiques au sens large, à l'aide d'appels aux diverses routines génériques.

Description :

- Initialisations effectuées lors du premier appel.
- Appel à l'Equation of State pour le calcul des grandeurs thermodynamiques.
- Détermination de l'énergie thermonucléaire.
- Détermination de l'opacité.
- Calcul du gradient radiatif.

- Formation du critère de convection (Schwarzschild ou Ledoux).
- Affectation du gradient adiabatique ou radiatif dans les extensions des zone convective, *cf.* §3.8 (Page 22).
- Pour les zones convectives, calcul du gradient.

Appel :
`thermo` est appelée de divers endroits en particulier par `lim_zc`, *cf.* §7.42 (Page 204), `static_m`, `static_r`, *cf.* §7.76 (Page 226).

```
SUBROUTINE thermo(pt,p,t,m,l,r,dlpp,xchim,dxchim,
     1 ro,drop,drot,drox,u,dup,dut,dux,grad,dgradpt,dgradp,dgradt,
     2 dgradx,dgradm,dgradl,dgradr,dgradlpp,
     3 gam,dgampt,dgamp,dgamt,dgamx,dgamm,dgaml,dgamr,dgamlpp,
     4 epsilon,depsp,depst,depsx,kap,dkapp,dkapt,dkapx,
     5 delta,deltap,deltat,deltax,cp,dcpp,dcpt,dcpx,
     6 gradad,dgradadp,dgradadt,dgradadx,
     7 hp,dhppt,dhpp,dhpt,dhpx,dhpr,dhpm,
     8 gradrad,alfa,beta,gamma1,radiatif)
```

- Entrées :
  - `pt, p, t, m, l, r`: Pression totale, pression gazeuse, température, masse, luminosité, rayon.
  - `xchim, dxchim, dlpp`: Composition chimique par mole, dérivée / $(M/M_\odot)$, $\frac{\mathrm{d}\ln P_{\rm gaz}}{\mathrm{d}\ln P_{\rm tot}}$.
- Sorties :
  - `ro, drop, drot, drox`: densité et dérivées;
  - `u, dup, dut, dux`: énergie interne et dérivées;
  - `grad, dgradpt, dgradp, dgradt, dgradx, dgradm, dgradl, dgradr, dgradlpp`: gradient de température $\nabla \equiv \frac{\partial \ln T}{\partial \ln P_{\rm tot}}$ et dérivées;
  - `gam, dgampt, dgamp, dgamt, dgamx, dgamm, dgaml, dgamr, dgamlpp`: efficacité de la convection et dérivées;
  - `epsilon, depsp, depst, depsx`: énergie nucléaire et gravifique et dérivées;
  - `kap, dkapp, dkapt, dkapx`: opacité et dérivées;
  - `delta, deltap, deltat, deltax`: δ et dérivées;
  - `cp, dcpp, dcpt, dcpx`: $c_{\rm p}$ et dérivées;
  - `gradad, dgradadp, dgradadt, dgradadx`: gradient adiabatique et dérivées;
  - `hp, dhppt, dhpp, dhpt, dhpx, dhpr, dhpm`: échelle de hauteur de pression et dérivées;
  - `gradrad, alfa, beta, gamma1, radiatif`: gradient radiatif, α, β, Γ_1, `radiatif`=.TRUE.: on est dans une zone radiative.

7.81 Routine `thermo_atm`

Pour la restitution de l'atmosphère, cette routine PRIVATE du module `mod_atm`, *cf.* §D.8 (Page 364), est l'homologue de la précédente. Elle n'en diffère que par :

- l'abscence d'appel aux taux des réactions thermonucléaires,
- le calcul du gradient de température qui tient compte de la loi $T(\tau)$ utilisée pour la restitution de l'atmosphère *cf.* §6.3.2 (Page 67),
- la composition chimique, la luminosité et la vitesse angulaire y sont spatialement constantes.

Description :

- Initialisations effectuées lors du premier appel.
- Appel à l'Equation of State pour le calcul des grandeurs thermodynamiques.
- Détermination de l'opacité.
- Calcul du gradient radiatif.
- Formation du critère de convection (Schwarzschild ou Ledoux).
- Pour les zones convectives, calcul du gradient.

Appel :
`thermo_atm` est appelée par `eq_atm`, *cf.* §7.10.1 (Page 165).

```
SUBROUTINE thermo_atm(pt,p,t,xchim,m,l,r,dlpp,
     1 tau,df_tau,d2f_tau,rstar,ro,drop,drot,kap,dkapp,dkapt,gradad,
     2 dgradadp,dgradadt,grad,dgradpt,dgradp,dgradt,dgradr,dgradrs,
     3 dgradm,dgradtau,dgradlpp,gam,dgampt,dgamp,dgamt,dgamr,dgamrs,
     4 dgamm,dgamtau,dgamlpp,hp,dhppt,dhpp,dhpt,dhpr,dhpm,delta,deltap,
     5 deltat,cp,gradrad,alfa,beta,gamma1,radiatif,deriv)
```

- Entrées :
  - `pt, p, t, m, l, r`: pression totale, pression gazeuse, température, masse, luminosité, rayon,
  - `xchim, dlpp`: composition chimique par mole, $\frac{\mathrm{d}\ln P_{\mathrm{gaz}}}{\mathrm{d}\ln P_{\mathrm{tot}}}$;
  - `tau, df_tau, d2f_tau, rstar`: profondeur optique, partie dépendante de τ de la loi $T(\tau)$ et dérivées premières et secondes, rayon total;
  - `deriv`=.TRUE.: le calcul des dérivées est requis.
- Sorties :
  - `ro, drop, drot`: densité et dérivées;
  - `grad, dgradpt, dgradp, dgradt, dgradr, dgradrs, dgradm, dgradtau, dgradr, dgradlpp`: gradient de température $\nabla \equiv \frac{\partial \ln T}{\partial \ln P_{\mathrm{tot}}}$ et dérivées;
  - `gam, dgampt, dgamp, dgamt, dgamr, dgamrs, dgamm, dgamtau, dgamlpp`: efficacité de la convection et dérivées;
  - `kap, dkapp, dkapt`: opacité et dérivées;
  - `gradad, dgradadp, dgradadt`: gradient adiabatique et dérivées;
  - `hp, dhppt, dhpp, dhpt, dhpr, dhpm`: échelle de hauteur de pression et dérivées;
  - `delta, deltap, deltat`: δ et dérivées;
  - `cp, gradrad, alfa, beta, gamma1, radiatif`: c_{p}, gradient radiatif, α, β, Γ_1, `radiatif`=.TRUE.: on est dans une zone radiative.

7.82 Routine trho

La fonction de cette routine PRIVATE du module mod_atm, *cf.* §D.8 (Page 364), est la lecture des données et l'interpolation en (T, ρ) des lois $T(\tau)$ de type roger**, *cf.* §?? (Page ??).

Description :
D'origine externe cette routine n'est pas décrite.

Appel :
trho est appelée des routines de lois $T(\tau)$ du type roger**.

7.83 Routine update

Suivant la valeur .TRUE. / .FALSE. de la variable logique next, cette routine PRIVATE du module mod_static, *cf.* §D.12 (Page 367), permet, soit de translater au temps t le modèle calculé pour le temps $t + dt$, soit de réinitialiser un modèle pour le temps $t + dt$ avec celui obtenu pour le temps t. L'extension _t permet de différencier les variables au temps t de celles au temps $t + dt$.

Description :

- Si next=.TRUE. les grandeurs caractérisant le modèle au temps $t + dt$ sont transcrites sur celles du modèle au temps t, le pas temporel est ajusté de façon à éventuellement arrêter le calcul sur l'âge maximum désiré.
- Si next=.FALSE. les grandeurs caractérisant le modèle au temps t sont transcrites sur celles du modèle au temps $t + dt$, le pas temporel est ajusté de façon à éventuellement arrêter le calcul sur l'âge maximum désiré.

Appel :
update est appelée par resout, *cf.* §7.67 (Page 219).

```
1 SUBROUTINE update(next,dt,dts)
```

- Entrées :
  - next=.TRUE.: On écrase les variables sur celles du pas temporel précédent.
  - next=.FALSE.: On restitue les variables du pas temporel précédent.
- Entrées/Sorties :
  - dt, dts: Pas temporel, pas temporel estimé.

7.83.1 Subroutine wind

```
subroutine wind( xchim, dxchim, jac )
```

Public subroutine of module mod_nuc.

Modifies the time derivatives of the chemical composition if the wind has a different chemical composition than the one of convective envelope.

wind is called only if l_wind=.True.

Arguments:

xchim : **real**(kind=dp), **optional**, **intent**(in), **dimension**(:)
Chemical composition.

dxchim : **real**(kind=dp), **optional**, **intent**(inout), **dimension**(:)
Modified time derivatives of the chemical composition.

jac : **real**(kind=dp), **optional**, **intent**(inout), **dimension**(:,:)
Jacobian.

Orignal author(s):

- P. Morel, Departement Cassiopee, O.C.A., CESAM2k.

When there is mass loss or gain, Cesam2k20 offers the possibility that the chemical composition of the gained or lost matter differs from that of the outermost layers (see Sect. 6.7.1).

During an initialization call from the evol routine, a search is made in the environment for the file indicating the mass fractions of the wind (see Sect. 3.15.4). If such a file is found, the mass fractions are normalized, then distributed among the different isotopes of the nuclear network, and finally transformed into mole fractions; the logical variable l_wind is declared .true.. If the file cannot be found, the logical variable l_wind is declared .false., and the chemical composition of the wind will be that of the outermost layer of the model. During subsequent calls, a correction to the nuclear reaction rates is made (see Sect. 6.7.1).

wind is called by:

- evol (see Sect 7.31.3), for initialization and in cases where the wind has a different chemical composition from that of the outermost layer.
- rk_imps (see Sect 7.71), or eq_dif_chim (see Sect 7.17), following a call to the nuc routine (see Sect 7.6.1).

7.84 Routine write_nl

Cette routine PUBLIC du module mod_exploit, *cf.* §D.14 (Page 368), a pour fonction la formation d'un fichier de données mon_modele.don, *cf.* §3.3 (Page 14).

Description :

Les NAMELISTs sont écrites après l'ouverture du fichier.

Appel :

write_nl, qui n'a pas d'argument, est utilisée par des programmes d'exploitation, *cf.* §?? (Page ??).

7.85 Package z14xcotrin21

Le package d'interpolation d'opacité z14xcotrin21 de A.I.Boothroyd, est utilisé par Cesam2k20 pour le calcul de l'opacité, lorsque l'abondance en éléments lourds dépasse $Z > 0.1$. Les routines d'interpolations utilisent des tables du sous-directory SUN_STAR_DATA, *cf.* §?? (Page ??). Elles nécessitent une mise en forme qui est transparente pour l'utilisateur *cf.* §?? (Page ??).

La mise en oeuvre du package z14xcotrin21 a bénéficié d'une collaboration avec A.I.Boothroyd. En raison de la complexité de sa programmation, z14xcotrin21 n'est pas inclus dans un module. Le package est compilé conjointement au module mod_opa, *cf.* §D.6 (Page 364).

Description :

En raison de leur complexité et de leur origine externe, les routines du package ne sont pas détaillées. Pour plus d'informations se reporter aux explications données en tête de la source de z14xcotrin21.

Appel :

Avec le package z14xcotrin21 Cesam2k20 utilise la routine opal_x_cno_fu ou à défaut la routine opal. Ces routines sont appelées par opa_opal2, *cf.* §7.53 (Page 211). En raison de la complexité, les listes d'appel ne sont pas détaillées.

7.86 Programmes cesam2k, cesam2k_dbg

Ces deux programmes permettent l'exploitation et le debug de Cesam2k20. Ils ne comportent que l'instruction d'appel à la routine cesam, *cf.* §7.7.3 (Page 158). Ils ne sont différenciés que par des options de compilation qui optimisent l'exécution ou permettent le debug. (Les deux programmes cesamT et cesamT_dbg identiques aux précédents et utilisés pour des tests, ne sont laissés dans la distribution que pour en faciliter la gestion.)

Chapter 8

Numerical subroutines

Contents

Ces routines sont des routines PUBLIC ou PRIVATE du module mod_numerique, *cf.* §D.2 (Page 353) ou encore mod_exploit, *cf.* §D.14 (Page 368). Elles relèvent de l'analyse numérique et de l'algorithmique. Certaines routines sont originales, d'autres ont été construites en s'inspirant des descriptions données dans des ouvrages d'analyse numérique élémentaire. On n'en donne qu'une description sommaire. Ces routines sont regroupées dans le sous-directory SOURCE.

8.1 Routines numériques et assimilées

8.1.1 Function adams_bashforth

```
function adams_bashforth(dt, vals, order) result(next_val)
```

Extrapolate a series of possibly 3 values with a Adams-Bashforth method of order 3, 2, 1 or 0.

Public subroutine of mod_numerique Extrapolate a series of possibly 3 values with a Adams-Bashforth method of order 3, 2, 1 or 0.

Arguments:

dt : **real**(kind=dp), **intent**(in), **dimension**(:)
Time steps.

vals : **real**(kind=dp), **intent**(in), **dimension**(:)
Values.

order : **integer**, **intent**(in)
Order of the extrapolation.

History:

2019-2 : Initial Code (Louis Manchon).

8.1.2 Function adams_bashforth_real

```
function adams_bashforth_real( this, constant ) result(next_val)
```

Determine which order ust be used for Adams-BAshforth exrapolation, give a list of 3 reals, Possibly NaN.

Public subroutine of mod_numerique Determine which order ust be used for Adams-Bashforth exrapolation, give a list of 3 reals, Possibly NaN. Calls adams_bashforth accordingly.

Arguments:

this : **class**(extrapolation_real), **intent**(inout)

constant : **logical**, **optional**, **intent**(in)
If true, returns value at previous time step when only first order scheme can be used.

History:

2019-2 : Initial Code (Louis Manchon).

8.1.3 Routine arb_rom

Cette fonction transforme l'entier $i \in [0, 30]$ en notation arabe en notation romaine, Exemple: $21 \Longrightarrow XXI$. La notation romaine comporte au plus 10 caractères, au delà, la sortie est CCC. Cette fonction est utilisée pour noter les niveaux d'ionisation.

8.1.4 Routine boite

Cette routine permet de dessiner une boîte d'erreur dont on donne les coordonnées du centre et la longueur des cotés.

8.1.5 Routine box

Cette routine permet de dessiner une boîte de dimensions $\pm\Delta x \times \Delta y$ autour du point de coordonnées (x, y).

8.1.6 Routine delete_doubles

Cette routine permet de supprimer les éléments successifs identiques dans un tableau ALLOCATABLE ordonné par ordre croissant; si besoin la dimension du tableau est réajustée.

8.1.7 Routine difdiv

Calcul des différences divisées pour interpolation par la formule de Newton *cf.* **?**.

8.1.8 Routine fermi_dirac

Approximations analytiques des intégrales de Fermi-Dirac, reprises du package MHD, *cf.* § 7.28.1 (Page 177).

8.1.9 Routine gauss_band

Routine de résolution d'un système linéaire bande, dont la largeur de la bande est constante et comportant plusieurs seconds membres. L'algorithme est celui de l'élimination de Gauss avec équilibrage et pivot partiel. L'espace mémoire utilisé correspond à la partie non identiquement nulle du système.

8.1.10 Subroutine gauss_leg

```
pure subroutine gauss_leg(x1, x2, x, w, n)
```

Subroutine of mod_num
Compute angle and weight from Gauss-Legendre quadrature method.

Arguments:

x1, x2 : **real**(kind=dp), **intent**(in)
Integration range $[x1, x2]$.

x : **real**(kind=dp), **intent**(out), **dimension**(n)
Abscissa.

w : **real**(kind=dp), **intent**(out), **dimension**(n)
Weights.

n : **integer**, **intent**(in)
Number of points in Gauss-Legendre quadrature.

References:

NumRes : (C) Copr. 1986-92 Numerical Recipes Software 23.D.

History:

2018-12 : Initial code from ACOR (Rhita-Maria Ouazzani).

8.1.11 Routine horner

La subroutine horner permet de calculer, pour $x = \alpha$, les dérivées de tous ordres d'un polynôme ainsi que le quotient de ce polynôme par $x - \alpha$ (algorithme de Hörner).

8.1.12 Function horner0

```
function horner0( n, a, x ) result( pol )
```

Public routine of mod_numerique. Horner algorithm without computation of derivatives.

Arguments:

n : **integer**, **intent**(in)
Order of te polynomial.

a : **real**(kind=dp), **intent**(in), **dimension**(n)
Coefficients of polynomial.

x : **real**(kind=dp), **intent**(in)
Evalution abscissa.

8.1.13 Routine intgauss

Initialisation des poids et des abscisses pour l'intégration de Gauss de divers ordres.

8.1.14 Routine matinv

Programme d'inversion de matrice. Utilise l'élimination de Gauss avec pivot total de gauss_band.

8.1.15 Routine max_local

Recherche des maxima de plusieurs tables dans un intervalle d'abscisses. Cette routine est adaptée au calcul des échelles pour des tracés.

8.1.16 Routine min_max

Recherche du maximum et du minimum d'une table. Cette routine est adaptée au calcul des échelles pour des tracés.

8.1.17 Routine min_max_cond

Recherche du maximum et du minimum d'une table sous condition. Cette routine est adaptée au calcul des échelles pour des tracés.

8.1.18 Routine neville

Algorithme de Neville pour interpolation polynômiale, *cf.* **?**.

8.1.19 Routine newton

Interpolation polynômiale avec dérivées, par la formule de Newton, *cf.* **?**.

8.1.20 Routine pause

Pause avec commentaire et poursuite/arrêt.

8.1.21 Routine pgplot_factice

Ensemble des intitulés des routines du logiciel PGPLOT utilisées dans Cesam2k20. Ces routines permettent d'exploiter le code sans que le logiciel de dessin soit implémenté, *cf.* § **??** (Page **??**). Cet ensemble est disposé à l'extérieur du MODULE mod_cesam.

8.1.22 Routine plot_rota

Routine subordonnée de ecrit_rota.f90, *cf.* § 7.20 (Page 173) Routine de dessin "on line" des variables de la diffusion du moment cinétique.

8.1.23 Routine polyder

C'est une autre version de l'algorithme de Hörner dont les arguments diffèrent de ceux de horner. Pour des raisons historiques Cesam2k20 fait appel aux deux versions.

8.1.24 Function pol2

```
real(kind=dp) function pol2( alpha, z )
```

Public routine of mod_numerique. Horner algorithm, for 2nd order polynomial, without computation of derivatives

Arguments:

alpha : real(kind=dp), intent(in), dimension(0:2)
Coefficients of polynomial.

z : real(kind=dp), intent(in)
Evalution abscissa.

8.1.25 Routine shell

Routine tri d'un tableau optimisé pour F95.

8.1.26 Routine zoning

Permet de déterminer la répartition des abscisses de façon à ce que les incréments des ordonnées soient constants.

8.2 Routines spécifiques aux B-splines

Ces routines sont originales dans leur grande majorité. Les algorithmes utilisés sont, pour la plupart, inspirés de ceux décrits par **??**.

8.2.1 Subroutine bsp1ddn

```
subroutine bsp1ddn( nf, f, x, xt, n, m, knot, init, xx, l, fx )
```

Public routine for the mod_mumerique module.

Interpolate in f(nf,n) arrays with a polynomial spline of order $m > 1$, at point xx : x(1) <= xx <= x(n) and xt(l) <= xx < xt(l+1) with calculation of all derivatives.

Take any l as input: $1 \leq l \leq n + 2m$ at first call, init=.false. there is initialisation: calculation of knot=2m+n, formation of the table xt(knot) of table points and f(n) coefficients of the B-splines.

Arguments:

xx : **real**(kind=dp), **intent**(in)
Interpolation point.

x : **real**(kind=dp), **intent**(in), **dimension** (:)
Abscissa.

m : **integer**, **intent**(in)
Order of the spline.

n : **integer**, **intent**(in)
Number of points.

nf : **integer**, **intent**(in)
Number of functions.

init : **logical**, **intent**(in)
If True, coefficients are already known.

f : **real**(kind=dp), **intent**(inout), **dimension**(:,:)
F(nf,n): functions to be interpolated / splines coefficients.

xt : **real**(kind=dp), **intent**(inout), **dimension** (:)
Xt(m+n) matching points.

knot : **integer**, **intent**(inout)
Knot = m+n: number of nodes.

l : **integer**, **intent**(inout)
Localisation.

fx : **real**(kind=dp), **intent**(out), **dimension**(:,0:)
Values of interpolated function. f(i,der) : derivative of order der for i -th function (e.g.: f(3,1) : 1st order derivative of 3rd function).

History:

1992-12-07 : F77 version (P. Morel, Departement J.D. Cassini, O.C.A).

2002-11 : Adapted toF95.

Generalises bsp1dn by calculating all non-identically zero derivatives. Should only be used if derivatives of order greater than unity are required, as it requires more resources than bsp1dn.

8.2.2 Routine bsp1dn

Interpolation au même point de n fonctions développées sur une base de B-splines; calcul du développement si nécessaire.

8.2.3 Extended type bspline2d

Subroutine assign_bspline2d

```
subroutine assign_bspline2d(this, other)
```

Define overloading of (=) for bspline2d
Function of bspline2d class. Define overloading of (=) for bspline2d

Arguments:

other : **class**(bspline), **intent**(in)

this : **class**(bspline2d), **intent**(out)

History:

2019-7 : First implementation (Louis Manchon).

Function bs_check_coherence2d

```
function bs_check_coherence2d(this) result(ok)
```

Check coherence of 2D spline.
Function of type_bspline2d. Adapted of bs_check_coherence1d written by Joao P. C. Marques

Arguments:

this : **class**(bspline2d), **intent**(in)
Instance of bspline2d class.

History:

2019-7 : Adaptation of bs_check_coherence1d.f90 (Louis Manchon).

8.2.4 Routine bsp_dis

Routine de calcul des coefficients des B-splines pour n fonctions avec discontinuités pour interpolations ou lissage par Bézier.

8.2.5 Routine bsp_gal

Routine de calcul des coefficients des B-splines pour n fonctions avec discontinuités pour interpolations; utilise le formalisme intégral de Galerkin.

8.2.6 Routine bval0

Routine qui calcule la valeur de toutes les B-splines non identiquement nulles en un point fixé, *cf.* § 6.1.1 (Page 51).

8.2.7 Routine bval1

Routine qui calcule la valeur de toutes les B-splines non identiquement nulles et leurs dérivées premières en un point fixé. Demande plus de ressources que bval0.

8.2.8 Routine bvald

Routine qui calcule la valeur de toutes les B-splines non identiquement nulles ainsi que toutes leurs dérivées en un point fixé. Demande plus de ressources que bval1.

8.2.9 Routine coll

Routine qui génère les points de collocation de la base de de Boor (1978) pour l'intégration des équations différentielles.

8.2.10 Fonction colpnt

Fonction calculant l'abscisse du i-ième point de collocation dans l'intervalle $]x(l), x(l+1)[$ pour la résolution d'une équation différentielle d'ordre r. colpnt est appelée par coll.

8.2.11 Routine left_right

Calcul des valeurs des n fonctions f et de leurs dérivées premières de part en un point quelconque d'une représentation par B-Spline. De part et d'autre d'un point du vecteur nodal ces valeurs peuvent différer, par exemple s'il y a une discontinuité.

8.2.12 Routine linf

Localisation d'un point dans une table d'abscisses en ordre strictement croissant, adapté aux B-splines.

8.2.13 Routine newspl

Effectue un changement de base de B-splines pour n polynômes par morceaux.

8.2.14 Routine newspl_gal

Effectue un changement de base de B-splines pour n polynômes par morceaux; utilise le formalisme intégral de Galerkin.

8.2.15 Routine noedif

Routine d'initialisation du réseau de points de table de la base de de Boor (1978), pour la résolution des équations différentielles par collocation.

8.2.16 Routine noein

Détermine la séquence de nœuds de raccord pour une interpolation "optimale" suivant de Boor (1978).

8.2.17 Routine noeu_dis

Détermine la séquence de nœuds de raccord pour une interpolation "optimale" en tenant compte de discontinuités; identique a noein s'il n'y a pas de discontinuité.

8.2.18 Routine noeud

Détermine la séquence de nœuds de raccord pour un vecteur des multiplicités donné.

8.2.19 Routine schu58_n

Calcul des valeurs en un point de n polynômes par morceaux à partir de leurs développements sur une base de B-splines et des valeurs des B-splines non identiquement nulles en ce point, *cf.* Schumaker (1981).

8.2.20 Routine sum_n

Calcule les intégrales de n splines entre des bornes données. Les coefficients doivent avoir été précédemment calculés par bsp1dn par exemple. Au premier appel, ces coefficients seront adaptés au calcul de l'intégrale, donc modifiés. Ils ne seront pas recalculés lors des appels ultérieurs. S'inspire de Schumaker (1981, alg. 5.19).

Part III

Documentation of the pycesam Python package

Chapter 9

Core package pycesam

Contents

9.1 Scripts

9.1.1 run_test.py

def set_alpha

def get_last_test

def convert_bool

def read_input

9.1.2 run_grid.py

class RunGridError

RunGridError.__init__(value):

no description provided

RunGridError.__str__():

no description provided

class RunParallel

RunParallel.__init__(models, params, fparams, finished_dir, not_finished_dir, second_try=False):

no description provided

RunParallel.run():

no description provided

RunParallel.make_folder(model_name, dest):

no description provided

RunParallel.try_move(ext, dest, name=None, files=None):

no description provided

RunParallel.move(model, dest, finish):

no description provided

RunParallel.remove_files(pre, suf):

no description provided

RunParallel.tidy_up(model, model_name):

no description provided

9.1.3 run_cesam2k20_parallel.py

def move

9.1.4 plot_struc.py

9.1.5 plot_osc.py

This manual page documents the plot_osc.py command line. plot_osc.py allows to display the values of various quantities inside several stellar models.

```
usage: plot_osc.py [-h] names [names...]
positional arguments:
  names       Names of the models
optional arguments:
  -h, --help  show this help message and exit
```

9.1.6 plot_kiel.py

9.1.7 plot_hr.py

def go_to_path

def on_close

9.1.8 convert2spins.py

class Convert2SPiNSError

Convert2SPiNSError.__init__(value):

no description provided

Convert2SPiNSError.__str__():

no description provided

def print_man

def process_model

9.1.9 calc_grid_freqs.py

class RunParallel

RunParallel.__init__(models, params, fparams, finished_dir, not_finished_dir):

no description provided

RunParallel.run():

no description provided

9.1.10 run_cesam2k20.py

This manual page documents the run_cesam2k20.py command line. run_cesam2k20.py allows to run a Cesam2k20 model, computes its frequency, run debug analyses, etc.

```
usage: run_cesam2k20.py [-h] [-d] [-l] [-f] [-c] [-n] [-v] [-C] [-p] [--pms
filename.pms] [--ciben CIBEN]
                        [--dt0 DT0] [-z] [--zams filename.zams] [-r] [--rep filename.rep]
                        model_name
positional arguments:
  model_name            Name of the model
optional arguments:
  -h, --help            show this help message and exit
  -d, --debug           Cesam2k20 is run in debug mode.
  -l, --log             all standard output of Cesam2k20 is stored in a.log file.
  -f, --freq           GUI is preset to frequency computation and no model computation.
  If used with
                        --nogui, automatically compute frequencies without computing
                        model.
  -c, --create          only create the don file without computing the model.
  -n, --nogui           immediatly compute what was asked, without opening GUI.
  -v, --valgrind        run Cesam2k20 with the Valgrind profiling tool
  (https://valgrind.org/).
  -C, --coverage        [not fully implemented yet] run Cesam2k20 and estimate code
  coverage with gcov
                        (https://gcc.gnu.org/onlinedocs/gcc/Gcov.html).
  -p                    preset run's option to a start from the PMS.
  --pms filename.pms    same as '-p' but here, you can give the name of a specific.pms
  file
  --ciben CIBEN         Value of Iben's constant.
  --dt0 DT0             Initial time step
  -z                    preset run's option to a start from the ZAMS.
  --zams filename.zams  same as '-z' but here, you can give the name of a specific.zams
  file
  -r                    preset run's option to a start from the previous model.
  --rep filename.rep    same as '-r' but here, you can give the name of a specific.rep
  file
```

class CSetupGUI

CSetupGUI.__init__():

no description provided

9.1.11 make_grid.py

This manual page documents the make_grid.py command line. make_grid.py allows to create Cesam2k20's input files for a grid of models. The grid points can be set regularly, following the Sobol

algorithm, or chosen using a uniorm distribution.

```
usage: make_grid.py [-h] [-c grid_name] [-s NUM] [-y] [-i] [-S index] [input_file]
[grid_name]
positional arguments:
  input_file            Name of the file that contains parameters varying in the grid.
  See input.in in
                        pycesam/python3 for an example
  grid_name             The generic name of your grid.
options:
  -h, --help            show this help message and exit
  -c grid_name, --clean grid_name
                        Give the generic name of the grid if you want to remove all don
                        files.
  -s NUM, --start NUM   Starting index for models in the grid.
  -y, --yes             Don't ask before creating grid input files.
  -i, --index           Only writes parameters of each model to a file and do not
  create.don files.
  -S index, --summary index
                        Read the index file of the grid and print out main information
                        out of it
```

class MakeGridError

MakeGridError.__init__(message):

Exception class for make_grid.

Arguments:

message : str
Error message.

MakeGridError.__str__():

String representation of MakeGridError.

Returns: str string representation of MakeGridError.

class SobolParams

SobolParams.__init__(names, values):

Class representing parameters in a Sobol grid.

Arguments:

names : list of strings
Names of the parameters.

values : dict
Parameter values at grid points.

class RandUnifParams

RandUnifParams.__init__(names, values):

Class representing parameters in a ranomly uniformed grid.

Arguments:

names : list of strings
Names of the parameters.

values : dict
Parameter values at grid points.

class CartParams

CartParams.__init__(names, values):

Class representing a parameter in a Cartesian grid.

Arguments:

names : list of strings
Names of the parameters.

values : list of numpy.ndarray
Parameter values at grid points.

class Params

Params.__init__(kind, names, values):

Generic Class representing a parameter in a any kind of grid.

Arguments:

kind : str
Type of grid. Existing kind: 'sobol', 'cartesian', 'rand_unif'.

names : list of strings
Names of the parameters.

values : dict or list of numpy.ndarray
Parameter values at grid points.

Params.__contains__(key):

Overrides in operator for class Params.

Arguments:

key : str
Checks if key is in the keys of `self.values`.

Returns: bool True if keys is in the keys of `self.values`.

Params.__iter__():

Defines iterator for the class Params.

Returns: 'dict_keyiterator' iterator of the self.values attribute.

Params.__len__():

Overides len function for class Params.

Returns: int Number of keys in self.values.

Params.__getitem__(key):

Overrides [] operator to get value for class Params.

Arguments:

key : str
key to be fetch in self.values dict.

Returns: np.ndarray Vaues at grid points for parameter key.

Params.__setitem__(new_key, new_val):

Overrides [] operator to set value for class Params.

Arguments:

new_key : str
key to be defined in self.values dict.

new_val : np.ndarray
value to be associated with a given key in self.values dict.

class Grid

Grid.__init__():

Class that reresents a grid of models.

Errors raised:

MakeGridError : If template.don does not exists.

Grid.__sobol_grid(param_in, ntracks, diffusion='No'):

Generates quasi-random (sobol) grid points in a specified parameter space (Private method).
Variable supported: 'mtot', '[Fe/H]0', 'y0', 'z0', 'alpha', 'ovshts', 'mdot', 'ovshti'

Arguments:

ntracks : int
Number of tracks

diffusion : str
Is diffusion taken into account ? If 'Yes', include diffusion for all tracks by setting the corresponding parameter to 1. If 'No', exclude diffusion for all tracks by setting the corresponding parameter to 0. If 'Mixed', certain tracks include diffusion and some do not, depending on the value of a quasi-random variable.

param_in : dict
Dictionnary containuing name of variable and min and max of the interval. Limited to 6 parameters at the moment.

Returns: dict Grid points

Errors raised:

MakeGridError : Diffusion parameter is not recognized.

Grid.__rand_unif_grid(param_in, ntracks):

Generates grid points big choosing randomly with a uniform distribution in a parameter range. (Private method).

Arguments:

param_in : dict
Dictionnary containuing name of variable and min and max of the interval. Limited to 6 parameters at the moment.

ntracks : int
Number of tracks

Returns: dict Grid points

Grid.__clean():

no description provided

Grid.__summary():

no description provided

Grid.arg_parse():

Argument parser for CLI.

Grid.make_grid():

Create grid input files.

Grid.extract_dYdZ(lines):

Extracts Constrains on dY/dZ.

Arguments:

lines : list of str
Lines in input file.

Grid.extract_ovs(lines):

Extracts Constrains on overshoot.

Arguments:

lines : list of str
Lines in input file.

Grid.extract_sobol(lines):

Extracts Sobol's requirements

Arguments:

lines : list of str
Lines in input file.

Errors raised:

MakeGridError(1) : You did not provided the number of tracks needed.
MakeGridError(2) : The `sobol_numbers` module is not availble.

Grid.extract_rand_unif(lines):

Extracts requirements for genereteaing a random uniform grid.

Arguments:

lines : list of str
Lines in input file.

Errors raised:

MakeGridError : You did not provided the number of tracks needed.

Grid.extract_meaningfull_lines():

Extracts un-commented lines in input file.

Returns: list of str Meaningfull lines.

Grid.make_cartesian_grid(lines):

Builds a cartesian (evenly spaced) grid.

Arguments:

lines : list of str
Lines in input file.

Returns: Class representing grid points.

Errors raised:

MakeGridError : You cannot have z0 and [Fe/H]0 in the input file at the ame time.

Grid.make_rand_unif_grid(lines):

Builds a random (uniform distribution) grid. Values at grid points are chosen randomly in an interval.

Arguments:

lines : list of str
Lines in input file.

Returns: Class representing grid points.

Errors raised:

MakeGridError : You cannot have z0 and [Fe/H]0 in the input file at the ame time.

Grid.make_sobol_grid(lines):

Builds a quasi-random grid. Values at grid points are chosen with a Sobol algorithm in an interval.

Arguments:

lines : list of str
Lines in input file.

Returns: Class representing grid points.

Errors raised:

MakeGridError(1) : You cannot have z0 and [Fe/H]0 in the input file at the ame time.
MakeGridError(2) : Current implemetation of the Sobol algorithm cannot handle more than 6 parameters.

Grid.feh_to_zsx():

Switches from [Fe/H] to z for sobol and randomly uniform grids.

Grid.clean_dYdZ(y0):

Removes the model that do not meet the constrain on dY/dZ.

Arguments:

y0 :
Initial helium content of the template model.

Returns: int Number of remaing model in the grid.

Grid.clean_models():

Selects models in the grid according to constraints set in the input file.

Grid.write_don_files():

Write.don files of models selected in the grid

9.1.12 constants.py

9.2 __init__.py

9.2.1 class CModel

CModel.__init__(name, reinit=False, read=True, job=None, fromGUI=False, verbose=True):

Class that represents a Cesam2k20 evolution model.

Arguments:

name : string
Name of the model.

Keyword arguments:

reinit : boolean
If True, deletes all calculated files. Default: False.

read : boolean
If True, reads files.HR and.osc. Default: True.

job : dict
Some options we want to enforce.

fromGUI : boolean
True if called from `super()` function in CModelGUI. Default: False

Errors raised:

NameError : if name length is 0 or > 31 characters.

CESAMError : If name of model starts with a hyphen '-'.

CModel.__str__():

Gives the string representation of a CModel instance.

Returns: string Mass, X0, Y0 and Z0 of the model if has been correctly calculated. Otherwise, only mass.

CModel.__repr__():

Gives the representation of a CModel instance (see `self.__str__`).

Returns: string Mass, X0, Y0 and Z0 of the model if has been correctly calculated. Otherwise, only mass.

CModel.__getitem__(key):

Return value of a given attributename.

Errors raised:

`KeyError` : if attribute does not exist in class.

CModel.__setitem__(key, value):

Set value to a given attributename.

Errors raised:

`KeyError` : if attribute does not exist in class.

CModel.__eq__(other):

Checks whether model parameters are equal to parameters of another model.

Arguments:

`other` :
Instance of class CModel.

CModel.__len__():

Returns number of time steps in the model

CModel.__ne__(other):

Checks whether model parameters are not equal to parameters of another model.

Arguments:

`other` :
Instance of class CModel.

CModel.__call__(mkdon=True, mkeos=True, debug=False, log=False, devnull=False, verbose=None, **kwargs):

Calculates models or frequencies.

Keyword arguments:

`mkdon` : boolean
Creates (or re-creates) a.don file if `mkdon==True` ; uses existing.don file otherwise. Default: True.

`mkeos` : boolean
Creates (or re-creates) the EoS file if `mkeos==True` ; uses existing EoS file otherwise. Default: True.

debug : boolean
If True, run the debug Cesam2k20 executable (`cesam2k20_dbg.x`). Default: False.

log : boolean
If True, direct standard output to.log file. Default: False.

devnull : boolean
If True, redirect standard output to /dev/null.

verbose : boolean
If True, information is written for the user. Default: True.

`**kwargs`:

valgrind : boolean
If True, calls Cesam2k20 with valgrind analysis software.

coverage : boolean
If True, evaluate test coverage by calling `cesam2k20_cov.x` with gcov.

CModel.__precompute_pms(valgrind=False, coverage=False, debug=False, log=False, devnull=False):

no description provided

CModel.__init():

Method used to initialize all subclasses used by CModel and read files if neaded.

CModel.__process_out(out, err, tstart, verbose, debug, log, valgrind):

Method that process the output of Cesam2k20 (Was the evolution a success or was it aborted?), and read the output files if needed

Arguments:

out : array of str
Contains all messages printed to standard output by Cesam2k20.

err : array of str
Contains all messages printed to standard error by Cesam2k20.

tsart : float
Time at which the computation started.

verbose : boolean
If True, information is written for the user. Default: True.

debug : boolean
If True, run the debug Cesam2k20 executable (`cesam2k20_dbg.x`). Default: False.

log : boolean
If True, direct standard output to.log file. Default: False.

valgrind : boolean
If True, calls Cesam2k20 with valgrind analysis software.

CModel.__tail(file, n=1):

no description provided

CModel.__run_cesam(debug=False, log=False, devnull=False, **kwargs):

Private method that runs one of the Cesam2k20 executable. If needed, it stores the standard output to file.

Keyword arguments:

debug : boolean
If True, run in debug mode (runs 'cesam2k20_dbg.x' instead of 'cesam2k20.x')

log : boolean
If True, stores standard output to file.

devnull : boolean
If True, redirect standard output to /dev/null.

**kwargs:

valgrind : boolean

coverage : boolean

CModel.is_finished():

Checks if the computation of the evolutionary track has been carried to the end. Modifies boolean attribute self.finished.

CModel._reinit():

Removes all calculated files. Does not remove file.don.

CModel.set_osc():

Find the names of.osc files; stores names in self.osc.

CModel.set_osc2d():

Find the names of.osc2d files; stores names in self.osc2d.

CModel.set_rep():

Find the names of.rep files; stores names in self.file_rep.

CModel.set_agsm():

Find the names of.agsm files; stores names in self.agsm.

CModel.set_acor():

Find the names of.acor files; stores names in self.acor.

CModel.set_amdls():

Find the names of.amdl files; stores names in self.amdls.

CModel.init_fparams(p_or_g, keep=False):

Initializes parameters for frequency calculations.

Arguments:

p_or_g : character
Can be either 'p' or 'g'.

Keyword arguments:

keep : bool
If True : keeps default parameters.

CModel.isoc(age, x):

Returns a variable x interpolated in age.

Arguments:

age : float
Desired age.

x : array or list
Table of values to interpolate. Must have the same size as self.age.

Returns: float Value of quantity x at desired age.

Example: To get log_teff at t = 20.0 Myrs for model m:

```
log_teff = m.isoc(20.0, m.log_teff)
```

CModel.read_hr(obs=False, zsx_sol=None, verbose=None):

no description provided

CModel.__read_hr_new(obs=False, zsx_sol=None, verbose=True):

Reads the model's HR file. Computes colors and visual magnitudes if required. Teff-color transformations and bolometric corrections from VandenBerg & Clem, 2003, AJ, 126, 778 (uses the program described by VandenBerg et al., 2006, ApJS, 162, 375).

Keyword arguments:

obs : bool
If True, computes colors and visual magnitudes. Default: False

zsx_sol : float
Solar Z/X. If None, will use solar Z/X correponding to model parameter 'Initial abundancies'. If unable, will use default value of 0.0181 (Asplund et al. 2009).

verbose : bool
If True, print some details.

New attributes:

age : np.ndarray
Age of each models.

log_teff : np.ndarray
$\log T_{\rm eff}$ of each models.

log_l : np.ndarray
$\log T_{\rm eff}$ of each models.
log_r : np.ndarray
$\log T_{\rm eff}$ of each models.
log_g : np.ndarray
$\log T_{\rm eff}$ of each models.
FeH : np.ndarray
[Fe/H] of each models.
z : np.ndarray
Metallicity of each models.
mstar : np.ndarray
$M_\star$ of each models.
omega_s : np.ndarray
Surface angular velocity of each models.
vrot : np.ndarray
Rotation velocity of each models.
alpha : np.ndarray
α parameter of each models.
senv : np.ndarray
Entropy of adiabat of each models.
mu : np.ndarray
Mean molecular weight at bottom of CZ of each models.

CModel.__read_hr_old(obs=False, zsx_sol=None, verbose=True):

Reads the model's HR file. Computes colors and visual magnitudes if required. Teff-color transformations and bolometric corrections from VandenBerg & Clem, 2003, AJ, 126, 778 (uses the program described by VandenBerg et al., 2006, ApJS, 162, 375).

Keyword arguments:

obs : bool
If True, computes colors and visual magnitudes. Default: False
zsx_sol : float
Solar Z/X. If None, will use solar Z/X correponding to model parameter 'Initial abundancies'. If unable, will use default value of 0.0181 (Asplund et al. 2009).
verbose : bool
If True, print some details.

New attributes:

age : np.ndarray
Age of each models.
log_teff : np.ndarray
$\log T_{\rm eff}$ of each models.
log_l : np.ndarray
$\log T_{\rm eff}$ of each models.
log_r : np.ndarray
$\log T_{\rm eff}$ of each models.
log_g : np.ndarray
$\log T_{\rm eff}$ of each models.

FeH : np.ndarray
[Fe/H] of each models.

z : np.ndarray
Metallicity of each models.

mstar : np.ndarray
$M_\star$ of each models.

omega_s : np.ndarray
Surface angular velocity of each models.

vrot : np.ndarray
Rotation velocity of each models.

alpha : np.ndarray
α parameter of each models.

senv : np.ndarray
Entropy of adiabat of each models.

mu : np.ndarray
Mean molecular weight at bottom of CZ of each models.

CModel.__close_hr(err=False, obs=False, verbose=True, old=False):

Properly close a HR file and remove data that have been partially read.

Keyword arguments:

err : boolean
If True, an error was found and partially read data must be removed.

obs : boolean
If True, some observable parameters are computed such as magnitude and colors.

verbose : boolean
If False, do not print details, except if err is True.

old : boolean
If True, HR file is in olf format

CModel.__process_cz():

Processe limits of convection zones, turning them into easier to plot quantities

New attributes:

m_conv : list
$m/M_\odot$ of each limit for each time step.

r_conv : list
$m/M_\odot$ of each limit for each time step.

iconv_start : list
index of the beginning of CZ.

iconv_end : list
index of the end of CZ.

mconv_start : list
lower mass of the CZ

rconv_start : list
lower radius of the CZ

mconv_end : list
upper mass of the CZ
rconv_end : list
upper radius of the CZ

CModel.__process_bz():

Processe limits of burning zones, turning them into easier to plot quantities

New attributes:

m_conv : list
$m/M_\odot$ of each limit for each time step.
r_conv : list
$m/M_\odot$ of each limit for each time step.
iconv_start : list
index of the beginning of CZ.
iconv_end : list
index of the end of CZ.
mconv_start : list
lower mass of the CZ
rconv_start : list
lower radius of the CZ
mconv_end : list
upper mass of the CZ
rconv_end : list
upper radius of the CZ

CModel.read_allrot(smooth=False):

Read all files `*_coeff_rota.dat`.

Keyword arguments:

smooth : bool
If True, `U2_t`, `V2_t` and `U_csi_t` are smoothed.

New attributes:

r_t : list of lists
Radius.
m_t : list of lists
Mass.
Omega_t : list of lists
Angular velocity.
U2_t : list of lists
Vertical meridional cirulation.
V2_t : list of lists
Horizontal meridional circulation.
Lambda_t : list of lists
Fluctuation of mean molecular weight.
Psi_t : list of lists
Fluactions of temperature.

Theta_t : list of lists
Fluctuation of density.

T_t : list of lists
Temperature.

U_csi_t : list of lists
$\frac{1}{2}\rho r^2 U_2$

CModel.read_rot2d(filename=None, old=False):

DEPRECIATED Reads the rorational coefficients file filename. If `filename=None` (default), reads `self.name+'_coeff_rota.dat2d'` ; if filename is an integer, reads file `number+'-*_coeff_rota.dat2d'`.

Errors raised:

TypeError : if type is not int or str. Sets: self.r self.m self.Omega self.U2 self.V2 self.Theta self.Upsi self.Lambda self.Deff self.Dh self.Dv self.T self.rho self.mu self.Psi

CModel.read_rot(filename=None, target_age=None, old=False):

Reads the rorational coefficients file filename.

Keyword arguments:

filename : string or int
Either a string corresponding to the name of the file, or an integer corresponding to the number of the time step (reads `number+'-*_coeff_rota.dat'`). If filename=None (default) and `target_age=None`, reads `self.name+'_coeff_rota.dat'`.

target_age : float
look of the model with closest age to `target_age`.

old : boolean
If True, the file is written in an old format.

Errors raised:

TypeError(1) : If filename is not int, str or float

TypeError(2) : If filename and `target_age` are None at the same time.

ValueError : If targeted age is higher than maximum age of the model

New attributes:

r : list
Radius.

m : list
Mass.

Omega : list
Angular velocity.

U2 : list
Vertical meridional cirulation.

V2 : list
Horizontal meridional circulation.

Lambda : list
Fluctuation of mean molecular weight.

Upsi : list
Value of Upsilon, see Cesam2k20's doc.

Psi : list
Fluactions of temperature.

Theta : list
Fluctuation of density.

Deff : list
Effective turbulent diffusion coefficient.

Dh : list
Horizontal turbulent diffusion coefficient.

Dv : list
Vertical turbulent diffusion coefficient.

T : list
Temperature.

rho : list
Density.

mu : list
Mean molecular weight.

U2s : list
Smoothed vertical meridional cirulation.

V2s : list
Smoothed horizontal meridional circulation.

U_csi : list
$\frac{1}{2}\rho r^2 U_2$.

U_csis : list
Smoothed U_csi.

CModel.j_ang(infile='rota', n_ini=None, n_end=None):

Calculates the evolution of total angular momentum.

Keyword arguments:

infile : string
Kind of infile file. Either 'rota' (default): _coeff_rota.dat file; or 'osc' : osc file.

n_ini : int
Starts computation from n_ini. Default: None.

n_end : int
Stops computation at n_end. Default: None.

CModel.read_osc_glob(filename):

Reads the glob part of an.osc file. Can be used to retrieve the value of the gravitational constant used to compute the model.

Arguments:

filename : string or list of string
Name of files to be read.

Returns: array of floats Global values

New attributes:

glob[0] : $M_\star M_\odot$

glob[1] : $R_{\rm tot} R_\odot$
glob[2] : $L_{\rm tot} L_\odot$
glob[3] : $Z0$
glob[4] : $X0$
glob[5] : α
glob[6] : X in CZ
glob[7] : Y in CZ
glob[8] : $\partial^2 p/\partial r^2$
glob[9] : $\partial^2 \rho/\partial r^2$
glob[10] : Age
glob[11] : $w_{\rm rot}$ initial (global rotation velocity)
glob[12] : $w_{\rm rot}$ initial
glob[13] : Gravitational constant
glob[14] : Solar mass
glob[15] : Solar radius
glob[16] : Solar luminosity
glob[17] : Entropy of the adiabat
glob[18] : Mean molecular weight at bottom of the adiabat

CModel.read_osc(filename=None, mods=None, read_vc=True, verbose=None, only_glob=False, fast=True):

Wrapper that reads the structure files. If `nom_output` is set to an HDF5 output, `read_osc` calls `read_osc_hdf5`, otherwise, it calls `read_osc_ascii` (except for `-plato.osc` files, it calls `read_osc_ascii_plato`).

Keyword arguments:

filename : string or list of string
Name of files to be read.

mods : list of integer
Index of the models to be read.

read_vc : boolean
If True, computes convective quantities (`vconv`, `Fconv`, `Pturb`...).

verbose : boolean
If True, print details.

only_glob : boolean
If True, only reads global parameters, not structure.

New attributes:

glob : array or list of array Global parameters for a model or for all models. Dimension: `nglob (, nmod)`

var : array or list of array Structure variables for a model or for all models. Dimension: `nvar, ntot (, nmod)` nvar = 22 for adiabatic oscillations

glob[0] : $M_\star M_\odot$
glob[1] : $R_{\rm tot} R_\odot$
glob[2] : $L_{\rm tot} L_\odot$
glob[3] : $Z0$
glob[4] : $X0$

glob[5] : α
glob[6] : X in CZ
glob[7] : Y in CZ
glob[8] : $\partial^2 p/\partial r^2$
glob[9] : $\partial^2 \rho/\partial r^2$
glob[10] : Age
glob[11] : $w_{\rm rot}$ initial (global rotation velocity)
glob[12] : $w_{\rm rot}$ initial
glob[13] : Gravitational constant
glob[14] : Solar mass
glob[15] : Solar radius
glob[16] : Solar luminosity
glob[17] : Entropy of the adiabat
glob[18] : Mean molecular weight at bottom of the adiabat
var[0,i] : $rR_\odot$
var[1,i] : $m/M_\star$
var[2,i] : T
var[3,i] : $P_{\rm tot}$
var[4,i] : ρ
var[5,i] : Actual gradient $dlnT/dlnP$
var[6,i] : L
var[7,i] : κ
var[8,i] : Thermal + gravitational energy
var[9,i] : Adiabatic index Γ_1
var[10,i] : Adiabatic gradient
var[11,i] : δ
var[12,i] : c_p
var[13,i] : $1/mu_{\rm elec.}$
var[14,i] : Brunt-Vaissala frequency, (0 at centre)
var[15,i] : Angular velocity [radian/sec]
var[16,i] : $d\ln\kappa/d\ln T$
var[17,i] : $d\ln\kappa/d\ln ro$
var[18,i] : $d\epsilon_{\rm nuc}/d\ln T$
var[19,i] : $d\epsilon_{\rm nuc}/d\ln ro$
var[20,i] : $P_{\rm tot}/P_{\rm gas}$
var[21,i] : Radiative gradient
var[22+j,i] : `xchim[j]*nucleo[j], j=0,nchim-1`

CModel.__read_osc_ascii(filename=None, mods=None, read_vc=True, verbose=True, only_glob=False):

Reads the file `*.osc`

Keyword arguments:

filename : string or list of string
Name of files to be read.

mods : list of integer
Index of the models to be read.

read_vc : boolean
If True, computes convective quantities (vconv, Fconv, Pturb...).

verbose : boolean
If True, print details.

only_glob : boolean
If True, only reads global parameters, not structure.

New attributes:

glob : array or list of array
See self.read_osc.

var : array or list of array
See self.read_osc.

CModel.__read_osc_ascii_fast(filename=None, mods=None, read_vc=True, verbose=True, only_glob=False):

Reads the file *.osc

Keyword arguments:

filename : string or list of string
Name of files to be read.

mods : list of integer
Index of the models to be read.

read_vc : boolean
If True, computes convective quantities (vconv, Fconv, Pturb...).

verbose : boolean
If True, print details.

only_glob : boolean
If True, only reads global parameters, not structure.

New attributes:

glob : array or list of array
See self.read_osc

var : array or list of array
See self.read_osc

CModel.__read_osc_ascii_plato(filename=None, mods=None, read_vc=True, verbose=True, only_glob=False):

Reads the file *.osc

Keyword arguments:

filename : string or list of string
Name of files to be read.

mods : list of integer
Index of the models to be read.

read_vc : boolean
If True, computes convective quantities (vconv, Fconv, Pturb...).

verbose : boolean
If True, print details.

only_glob : boolean
If True, only reads global parameters, not structure.

New attributes:

glob : array or list of array
See self.read_osc

var : array or list of array
Structure variables for a model or for all models.

var[0,i] : $rR_\odot$

var[1,i] : $m/M_\star$

var[2,i] : $P_{\rm tot}$

var[3,i] : ρ

var[4,i] : Adiabatic index Γ_1

var[5,i] : Brunt-Vaissala frequency, (0 at centre)

var[6,i] : Y

var[7,i] : Z

CModel.__read_osc_hdf5(filename=None, mods=None, read_vc=True, verbose=True, only_glob=False):

Reads the file *.osch5.

Keyword arguments:

filename : string or list of string
Name of files to be read.

mods : list of integer
Index of the models to be read.

read_vc : boolean
If True, computes convective quantities (vconv, Fconv, Pturb...).

verbose : boolean
If True, print details.

only_glob : boolean
If True, only reads global parameters, not structure.

New attributes:

glob : array or list of array See self.read_osc

var : array or list of array See self.read_osc

CModel._read_record(f_osc):

Private method that reads a record from a fortran binary file.

Arguments:

f_osc : '_io.TextIOWrapper'
Instance of an opened fortran binary file.

CModel.read_osc2d(filename, verbose=None):

Reads a osc2d file. For most 2D arrays, first dimension is the angular sector, 2nd dimension is the radius. For the angle, index goes from 0 to Nt (pole to equator).

Arguments:

filename : string
Name of files to be read.

Keyword arguments:

verbose : bool
If True, print out some comments

New attributes:

nph : int
Index of layer corresponding to the surface

Nr : int
Number of layers in the total structure (interior + atmosphere)

Nt : int
number of angular sectors

ml : int
number of legendre poly in the decomposition

ith : int
index of characteristic angle

cte_p : float
normalization cste for pressure ($p_{\rm file} = p_{\rm mod} cte_p$)

cte_rho : float
normalization cste for density ($\rho_{\rm file} = \rho_{\rm mod} cte_\rho$)

omega_k : float
break up velocity, normalization cste for omega ($omega_{\rm file} = omega_{\rm mod}/omega_{\rm k}$)

Rph : float
radius at the surface (at equator)

Req : float
radius with atm (at equator)

M : float
total mass

G : float
Universal gravity cste

theta : np.ndarray
angles of sectors

weight : np.ndarray
weight for legendre quadrature

cos_theta : np.ndarray
cos of angles

r2d : np.ndarray
radius of characteristics (spherical near center) (shape: `(Nt, Nr)`). In unit of `Req`

pr2d : np.ndarray
pressure on characteristics (shape: `(Nt, Nr)`)

rho2d : np.ndarray
density on characteristics (shape: `(Nt, Nr)`)

omega2d : np.ndarray
rotation on characteristics (shape: `(Nt, Nr)`)

gamma12d : np.ndarray
$Gamma_1$ on characteristics (shape: `(Nt, Nr)`)

drdz : np.ndarray
$dr2d/d\zeta$ (zeta is radius along characteristic angles) (shape: `(Nt, Nr)`)

dpdr : np.ndarray
dp/dr (shape: `(Nt, Nr)`)

drhodr : np.ndarray
$d\rho/dr$ (shape: `(Nt, Nr)`)

domegadr : np.ndarray
$d\Omega/dr$ (shape: `(Nt, Nr)`)

drdcos : np.ndarray
$dr/d\cos\theta$

dpdcos : np.ndarray
$dp/d\cos\theta$

drhodcos : np.ndarray
$drho/d\cos\theta$

domegadcos : np.ndarray
$dOmega/d\cos\theta$

d2rdz2 : np.ndarray
$d^2r/d\zeta^2$

d2rdcosdz : np.ndarray
$d^2r/d\cos\theta/d\zeta$

d2rdcos2 : np.ndarray
$d^2r/d\cos^2\theta$

d2pdcos2 : np.ndarray
$d^2p/d\cos^2\theta$

d2rhodcos2 : np.ndarray
$d^2\rho/d\cos^2\theta$

Ar : np.ndarray
$(dp/dr)/\Gamma_{1,2D}/pr2d - (d\rho/dr)/rho2d$

m : np.ndarray
Mass inside characteristic shells

rho_l : np.ndarray
component of the decomposition of rho over ISOBARS (only if option `extended_osc2d` was activated)

fp : np.ndarray
factor f_p (only if option `extended_osc2d` was activated)

ft : np.ndarray
factor f_t (only if option `extended_osc2d` was activated)

fd : np.ndarray
factor f_d (only if option `extended_osc2d` was activated)

Sp : np.ndarray
Surface of isobars (only if option `extended_osc2d` was activated)

geffav[0, : :] : np.ndarray
:] : $< g_{\text{eff}} >$ (only if option `extended_osc2d` was activated)

geffav[1, : :] : np.ndarray
:] : $< g_{\text{eff}}^{-1} >$ (only if option `extended_osc2d` was activated)

geffav[2, : :] : np.ndarray
:] : $< g_{\text{eff}}^{-1} r^2 \sin^2\theta >$ (only if option `extended_osc2d` was activated)

geff2d[0, : :, :] : np.ndarray
:, :]: $|g_{\rm eff}|$ (only if option `extended_osc2d` was activated)

geff2d[1, : :, :] : np.ndarray
:, :]: $g_{{\rm eff},{\rm r}}$ (only if option `extended_osc2d` was activated)

geff2d[2, : :, :] : np.ndarray
:, :]: $g_{{\rm eff},\theta}$ (only if option `extended_osc2d` was activated)

theta_m : np.ndarray
θ_m as a function of mass

U_l : nd.array
Legendre components of the vertical merid. circu. (only if transport by Mathis & Zahn 2004) in 2D is used)

V_l : nd.array
Legendre components of the horizontal merid. circu (only if transport by Mathis & Zahn 2004) in 2D is used)

Ups_l : nd.array
Legendre components of Upsilon (see doc.) (only if transport by Mathis & Zahn 2004) in 2D is used)

psi_l : nd.array
Legendre components of Psi (T_l/T) (only if transport by Mathis & Zahn 2004) in 2D is used)

lambda_l : nd.array
Legendre components of Lambda (μ_l/μ) (only if transport by Mathis & Zahn 2004) in 2D is used)

deff : nd.array
Effective diff. coeff for chemicals (e.g. Maeder 2003) (only if transport by Mathis & Zahn 2004) in 2D is used)

dh : nd.array
horizontal shear-induced diffusion coefficient (only if transport by Mathis & Zahn 2004) in 2D is used)

dv : nd.array
vertical shear-induced diffusion coefficient (only if transport by Mathis & Zahn 2004) in 2D is used)

omega_c : cut-off frequency (in units of `omega_k`)

omega_c_Hz : cut-off frequency in microHz

CModel.calc_vconv(origin, l_hp=True, K0=1.7, Gphi=2, beta=0.05, i=None, var=None, glob=None):

Computes the convective velocity, flux and the entropy fluctuations according to a given formulation of convection.

Arguments:

origin : str
Origin of the data. Either from osc file or rot file.

Keyword arguments:

l_hp : float
if True, mixing length = αH_p .

Gphi : float
Gough's anisotropy factor (default `Gphi=2`).

K0 : float
coefficient K0 (see H02), (default K0=1.5).

i : integer
Index of the time step for which we want to compute convective quantities.

var : 2d array
structure data of a model (single time step).

glob : 2d array
global data of a model (single time step).

Errors raised:

ValueError : You must either i or var and glob.

TypeError : Keyword i must be an integer.

Example: To compute convective quantities of model number 100, from data stored in an osc file:

```
mdl = CModel( 'model' )
mdl.calc_vconv( 'osc', i=100 )
mdl.calc_vconv( 'osc', i=-1 ) # for only last model
```

New attributes:

Fconv : np.ndarray
Convectif flux

Frad : np.ndarray
Radiatif flux

lmix : np.ndarray
Mixing-length (length $= \alpha H_p$)

Pturb : np.ndarray
Turbulent pressure

vconv : np.ndarray
Convective velocity

References:

H02 : Heiter et al, A&A, 2002

CGM : Canuto Goldman Mazitelli ApJ 473, 550-559, 1996

CM : Canuto Mazitelli ApJ 370, 295, 1991

PaperI : Samadi & Goupil, 2001

History:

28.04.03 : Reza Samadi

30.04.04 : Updated

11.02.11 : Adapted JP

CModel.write_osc(filename):

Write an osc file.

Arguments:

filename : string
Name of the osc file to be written.

CModel.calc_freqs(oscs=None, amdls=None, mods=None, write_amdl=True, quiet=False, read=True, from_gui=False, out_name=None):

Wrapper for the computation of frequencies that will either call ADIPLS or ACOR.

Keyword arguments:

oscs : list of strings
.osc files for which we want to compute the frequencies. oscs and amdls cannot be not None at the same time. If oscs and amdls are None, will compute frequencies for all.osc files.

amdls : list of strings
.amdl files for which we want to compute the frequencies. oscs and amdls cannot be not None at the same time. If oscs and amdls are None, will compute frequencies for all.osc files.

mods : list of integers
Indices of the time steps for which frequencies must be computed.

write_amdl : boolean
If True,.amdl have already been computed and we do not need to compute them again.

quiet : boolean
If True, standard output is not printed to screen.

read : boolean
If True, frequencies are read after computation.

from_gui : boolean
If True, `calc_freqs` was called from GUI. Default: False.

out_name : str
Name of the output.agsm file (useful when you compute only one file). If not given, the filename is constructed using the osc file. Default: None

CModel.calc_freqs_adipls(oscs=None, amdls=None, mods=None, write_amdl=True, read=True, from_gui=False, out_name=None):

Computes frequencies using the ADIPLS oscillation code (J. Christensen-Dalsgaard et al., 2008).

Keyword arguments:

oscs : list of strings
.osc files for which we want to compute the frequencies. oscs and amdls cannot be not None at the same time. If oscs and amdls are None, will compute frequencies for all.osc files.

amdls : list of strings
.amdl files for which we want to compute the frequencies. oscs and amdls cannot be not None at the same time. If oscs and amdls are None, will compute frequencies for all.osc files.

mods : list of integers
Indices of the time steps for which frequencies must be computed.

write_amdl : boolean
If True,.amdl have already been computed and we do not need to compute them again.

read : boolean
If True, frequencies are read after computation.

from_gui : boolean
If True, `calc_freqs` was called from GUI. Default: False.

out_name : str
Name of the output.agsm file (useful when you compute only one file). If not given, the filename is constructed using the osc file. Default: None

CModel.calc_freqs_acor(mods=None, quiet=False):

Computes frequencies using the ACOR oscillation code (R.-M. Ouazzani et al., 2015).

Keyword arguments:

mods : list of integers
Indices of the time steps for which frequencies must be computed.

read : boolean
If True, frequencies are read after computation.

CModel.calc_freq_lmax_acor(evol_code, nmod, d2, quiet=True):

Computes frequencies using the ACOR oscillation code (R.-M. Ouazzani et al., 2015), up to a certain degree lmax. ACOR can only find modes with a given parity between l and m. Therfore, in general, we need to run ACOR twice for even and odd (l,m). The difficulty is to find the number of needed spherical harmonics giving l and m. There may be a more elegant way than what I did here.

Arguments:

evol_code : string
Name of the stellar evolution code that was used to compute the structure model.

nmod : string
Name of the model that contains the structure.

d2 : boolean
If True, the structure model is a 2D model.

Keyword arguments:

quiet : boolean
If True, standard output will not be printed when frequencies are computed.

CModel.osc_amdl(osc=None):

Wrapper for the Fortran executable that converts an osc file to an amdl file.

Keyword arguments:

osc : string
Name of the osc file to be converted.

CModel.osc_losc(osc=None):

Wrapper for the Fortran executable that converts an osc file to an losc file.

Keyword arguments:

osc : string
Name of the osc file to be converted.

CModel.redistrb(amdl, out_name=None):

Runs the ADIPLS redistribution algorithm.

Arguments:

amdl : string
Name of.amdl file.

Keyword arguments:

out_name : string
If given, the amdl file is not overwritten. Default: None.

CModel.remesh(rep=None, don=None):

Remesh a model with points at a suitable location for accurate computation of p-modes ('p'), g-modes ('g'), modes in RGB stars ('r'), customized settings ('o').

Keyword arguments:

rep : string
Name of the rep file of the model that needs to be remeshed.

don : string
Name of the don file that was used to compute the model.

CModel.__setexec(nmod=None):

no description provided

CModel.__selsum(filein, fileout_gsm, fileout_ssm, n1, n2):

Private wrapper that calls ADIPLS's selsum program. It select desired modes in a summary file.

Arguments:

filein : str
Input file that contains modes characteristics.

fileout_gsm : str
File name to output grand summary.

fileout_ssm : str
File name to output short summary.

n1 : intint
Minimum order of in the selected modes set.Maximum order of in the selected modes set.

CModel.run_adipls(nn, nmod=None, ggrav=ggrav):

Wrapper of CModel.__run_adipls for public use.

Arguments:

nn :
no description provided

Keyword arguments:

nmod : string
Name of the stellar track.

ggrav : float
Gravitational constant to be used. Default: uses the globally defined ggrav.

CModel.__run_adipls(nn, nmod=None, ggrav=ggrav):

Private method called internally to run ADIPLS oscillation computations.

Arguments:

nn :
no description provided

Keyword arguments:

nmod : string
Name of the stellar track.

ggrav : float
Gravitational constant to be used. Default: uses the globally defined ggrav.

CModel.__run_acor(evol_code="CESAM2k20", nmod=None, d2=False, aparam_name='ACOR_parameters', quiet=False, procs=None):

Private method called internally to run ACOR oscillation computations.

Keyword arguments:

evol_code : string
Name of the evolution code that computed the structure model.

nmod : string
Name of the model that contains the structure.

d2 : boolean
If True, the structure model is a 2D model.

aparam_name : string
Name of the ACOR parameter file. Default: 'ACOR_parameters'.

quiet : boolean
If True, standard output will not be printed when frequencies are computed.

procs : list of CRunAcor instances
List containing the ACOR instances that need to be ran in parallel.

CModel.read_agsm(hdf5=False, filename=None, mods=None, dnu=True):

Wrapper that reads agsm files, either in classical binary format, or in an HDF5 file.

Keyword arguments:

hdf5 : boolean
If True, agsm file is an HDF5 file. Default False

filename : string
Name of the file we want to read.

mods : list of integers
Indices of the agsm files to be read.

dnu : boolean
If True, frequencies are post-processed, i.e. we compute frequency ratios, large and small seperation, etc. Default: True.

CModel.read_agsm_original(filename=None, mods=None, dnu=True):

Read frequencies in an original binary agsm file.

Keyword arguments:

filename : string
Name of the file we want to read.

mods : list of integers or list of strings
Indices of the agsm files to be read.

dnu : boolean
If True, frequencies are post-processed, i.e. we compute frequency ratios, large and small seperation, etc. Default: True.

CModel.read_agsmh5(filename=None, mods=None, dnu=True):

Read frequencies in an HDF5 file.

Keyword arguments:

filename : string
Name of the file we want to read.

mods : list of integers
Indices of the agsm files to be read.

dnu : boolean
If True, frequencies are post-processed, i.e. we compute frequency ratios, large and small seperation, etc. Default: True.

CModel.process_freqs():

Compute data products from frequencies such as frequency ratios, large and small seperation, etc.

CModel.read_acor(filename, suf='1d'):

Reads a reconstructed ACOR frequency file.

Arguments:

filename : string
Name of the model.

Keyword arguments:

suf : string
Suffix appended to the filename. Default: '1d'.

CModel.clean_freq_acor(cfacor):

Removes the modes that have been found several times.

Arguments:

cfacor : Instance of class CFacor
Instance of class CFacor that represents the frequencies.

CModel.__duplicate_mode(data, minoccur=2):

Find duplicate modes in acor frequency files.

Arguments:

data : 2d array
Array of integers n and l.

Keyword arguments:

minoccur : integer
Threshold at which an tuple (n,l) is considered to be a duplicate.

CModel.write_acor_freq(filename, cfacor):

Write acor frequencies in a file.acor.

Arguments:

filename : string
Name of thde frequency file.

cfacor : CFacor instance
CFacor instance representing acor frequencies.

CModel.pfreqs(p_or_g, nmod=-1):

no description provided

CModel.read_amdl(filename=None):

Reads an amdl file. Stores the result in `self.file_data[8]` and `self.aa[6,npoints]`.

Keyword arguments:

filename : str
Name of th file to be read. If no filename is specified, reads `self.amdl`.

CModel.read_amde(l, n, freq=None, filename=None, mod=None):

Reads an amde file. Reads eigenfunctions for given l and n from file filename

Arguments:

l : int
Degree of the mode.

n : int
Radial order of the mode.

Keyword arguments:

freq : float
If given, look for mode with closest frequency to freq. Default: None

filename : str
Filename in which data should be read. If None, look for `mod-name.amde` if mod is given and otherwise, for `self.amde`. Default None.

mod : int
Number of the time step. Default: None.

New attributes:

xeig : np.ndarray
Dimension: npoints

yeig : np.ndarray
Dimension: 6, npoints.

yeig[0] : ξ_r/R for radial modes or ξ_r/R for non-radial modes.

yeig[1] : $p'/(w^2R^2\rho)$ for radial modes or $l(l+1)\xi_h/R$ for non-radial modes.

yeig[2] : $(4\pi r^3\rho/M)^(1/2) * self.yeig[0]$ for radial modes or $-x\phi'/(gr)$ for non-radial modes.

yeig[3] : $x^2 d/dx(self.yeig[2]/x)$ for non-radial modes.

yeig[4] : $(4\pi r^3\rho/M)^(1/2) * self.yeig[0]$ for non-radial modes.

yeig[5] : $(4\pi r^3\rho/M)^(1/2)/\sqrt{l(l+1)} * self.yeig[1]$ for non-radial modes.

CModel.read_rotkr(l, n, freq=None, filename=None, mod=None):

Reads an rotational kernel file.

CModel.init_s(grid='original'):

Stores prescription parameters.

Keyword arguments:

grid : string
Grid on which the coefficients are calibrated, Either 'original' (default), 'cifist', 'cif_mdw' or 'cif_red' (recommended).

Errors raised:

ValueError : If law does not matched one of the available option.

CModel.fmu(mu1d, murhd, srhd=None):

no description provided

CModel.get_ds(law, grid='original', i=-1, fmu_simple=True):

Compute the entropy offset between a given prescription on a given grid, and the current model.

Arguments:

law : string
Name of the prescription. Either 'ludwig', 'tanner' or 'magic' (recommended).

Keyword arguments:

grid : string
Grid on which the coefficients are calibrated, Either 'original' (default), 'cifist', 'cif_mdw' or 'cif_red' (recommended).

i : integer
index of the model taken as reference (default: last, i=-1).

Returns: float Entropy offset.

Errors raised:

CESAMError : If you asked to use a model that was not computed or not used.

ValueError : If law does not matched one of the available option.

CModel.get_N2():

Computes the value of the Brunt-Vaisala frequencies for all time step associated to the model.

CModel.get_I(imod=None, imin=0, imax=-1):

Computes moment of inertia for all time step associated to the model.

Keyword arguments:

imin : int

imax : int

CModel.get_J(imod=None, imin=0, imax=-1):

Computes angular momentum for all time step associated to the model.

Keyword arguments:

imin : int

imax : int

9.3 freqs.py

9.3.1 class Freqs

Freqs.__init__(name, all_osc, verbose=True):

Class that represents ADIPLS or ACOR input parameters

Arguments:

name : str
Generic name of the model

all_osc : boolean
True if all osc files are written.

Keyword arguments:

verbose : dict
If True, prints additional information.

Freqs.manage_changes():

Adapts settings when the value of `self.modes` changes.

Freqs.l_changes():

Adapts settings when the value of `self.lmax`, `self.lmin` or `self.dels` changes.

Freqs.set_cts():

Adapts remesh settings when `self.remesh` changes

Freqs._fctsbc_changed():

Method automatically called whenever value of `self.fctsbc` is changed

Freqs._istsbc_changed():

Method automatically called whenever value of `self.istsbc` is changed

Freqs.write_fsettings():

Writes frequency settings to a `.frun` file.

Freqs.read_fsettings():

Reads frequency settings from a `.frun` file.

9.3.2 class CFacor

CFacor.__init__(filename=None):

Class that defines a data structure for ACOR's oscillation outputs.

Keyword arguments:

filename : str
Name of the file in which oscillation results are stored.

CFacor.merge(other_cfacor):

Merges a secondary `CFacor` instance to current instance. In practice: merges the two attributes `self.__dict__`.

Arguments:

other_cfacor :
Instance of `CFacor` whose attributes must be injected into current instance.

9.3.3 class CRunAcor

CRunAcor.__init__(fparams, evol_code="CESAM2k20", nmod=None, d2=False, aparam_name='ACOR_parameters', write=True, quiet=False):

Class that allows to run two instances of ACOR in parallel. Useful when we compute frequencies with odd/even parities.

Arguments:

fparams :
Instance of Freqs class that represents frequency parameters.

evol_code : stringnselsse
Name of the evolution code that computed the structure model. Optional.

nmod : string
Name of the model that contains the structure. Optional.

d2 : boolean
If True, the structure model is a 2D model. Optional.

aparam_name : string
Name of the ACOR parameter file. Default: `'ACOR_parameters'`.

write : boolean
If True, the ACOR parameter file is written when instance is created. Optional.

quiet : boolean
If True, standard output will not be printed when frequencies are computed. Optional.

CRunAcor.__write_aparam():

Private method that writes the file with ACOR parameters.

CRunAcor.run():

Run `script_run_ACOR_unified` executable. `CRunAcor.run` is only executed by the Thread class

CRunAcor.clean():

Clean directory of not needed files.

CRunAcor.read_acor_liste_freq():

Reads a `acor_liste_freq` file.

9.4 FortranIO.py

9.4.1 def myFromstring

9.4.2 class FortranBinaryFile

FortranBinaryFile.__init__(filename, mode='r', endian='@', verbose=0):

Fortran binary file support It is assumed that the file consists of Fortran records only. Constructor: FortranBinaryFile(filename, mode)

Arguments:

filename : str
Name of the file to be read.

mode :
no description provided

FortranBinaryFile.__del__():

no description provided

FortranBinaryFile.close():

no description provided

FortranBinaryFile.flush():

no description provided

FortranBinaryFile.readRecordNative(dtype=None):

no description provided

FortranBinaryFile.readRecordByteswapped(dtype=None):

no description provided

FortranBinaryFile.readBytes(recordsize, dtype, offset=None):

no description provided

9.5 constants.py

9.6 ces_tools.py

9.6.1 class RunParallel

RunParallel.__init__(model):

no description provided

RunParallel.run():

no description provided

9.6.2 def find_models

9.6.3 def run_parallel

9.7 tools.py

9.7.1 def get_hash

9.7.2 class Data

Data.__init__(header):

no description provided

9.7.3 class CESAMError

CESAMError.__init__(message):

no description provided

9.7.4 class CESAMGUIError

CESAMGUIError.__init__(message):

no description provided

9.7.5 def savitzky_golay

9.7.6 def isbool

9.7.7 def str2bool

9.7.8 def str2list

9.7.9 def str2type

9.7.10 def swap

9.7.11 class AlarmException

9.7.12 def alarmHandler

9.7.13 def nonBlockingInput

9.7.14 def email_attachement

9.7.15 def email

9.8 cparams.py

9.8.1 class CParameters

CParameters.__init__(name, verbose=True):

Class that represents the input parameters of a Cesam2k20 model, to be manipulated through GUI.

Arguments:

name : string
Name of the model.

Keyword arguments:

verbose : dict
If True, prints additional information.

CParameters.__dict_keys(dct, value):

Fetch the key associated to a value in a dictionnary of dictionnaries.

Arguments:

dct : dict of dict
Dictionary of dictionnaries form which the key should be fetched.

value : any object
Value that correspond to the kwy that should be fetched.

CParameters.__make_eos(curr_dir, eosfile, exec_path):

Compute new EoS table.

Arguments:

curr_dir : string
Directory where the computation of the model should take place.

eosfile : string
Name of the EoS file.

exec_path : string
Path to `make_eos5.x` executable.

CParameters.__process_value(name):

no description provided

CParameters._nom_chemin_changed():

Automatically every time `nom_chemin` is changed.

Errors raised:

CESAMError : Length should not exceed 120 characters.

CParameters._nom_pertm_changed():

no description provided

CParameters._x0_changed():

no description provided

CParameters._y0_changed():

no description provided

CParameters._z0_changed():

no description provided

CParameters._zsx0_changed():

no description provided

CParameters.change_nom_output():

no description provided

CParameters._trunc_nom_output_changed():

no description provided

CParameters._out_2d_changed():

no description provided

CParameters._out_h5_changed():

no description provided

CParameters.init_params():

no description provided

CParameters.copy(other):

no description provided

CParameters.read_params():

Reads a.don file so to obtain model parameters and physical inputs.

CParameters.mkdon(overwrite=True, replace_eos=True, verbose=True, eos_inplace=False):

Creates a.don file with the model parameters. Creates an EoS OPAL file for the right metalicity.

Arguments:

overwrite : boolean
If True, overwrites an existing file (default: True, optional)

replace_eos : boolean
If True, overwrites EoS OPAL file (default: True, optional).

verbose : boolean
If True, print some details (default: True, optional).

eos_inplace : boolean
If True, the eos file is stored in the folder where the computation tkes place. (default: True, optional)

9.9 cesam_run.py

9.9.1 class CRun

CRun.__init__(name, job=None, verbose=True):

Initialise options to control Cesam2k20's run.

Arguments:

name : str
Name of the model.

Keyword arguments:

job : dict
OPtion to set the starting point from elsewhere than the.run file.

verbose : dict
If True, prints additional information.

CRun.manage_mod_init_changes():

Automatically called by a decorator of `traits` module, any time `mod_init` is changed.

CRun.manage_type_file_changes():

Automatically called by a decorator of `traits` module, any time `type_file` is changed.

CRun.write_rsettings():

Write run settings to a.run file.

CRun.copy(other):

Copy run settings to another CRun instance, possibly from another model.

Arguments:

other :
Other instance of CRun, which have same parameters as `self`.

CRun.read_rsettings():

Read run settings in a.run file.

Chapter 10

Graphic User Interface pycesam.gui

Contents

10.1 __init__.py

10.1.1 class CModelGUI

CModelGUI.__init__(name, reinit=False, read=True, obs=False, freq=False, create=False, job=None):

Class that represent a Cesam2k20 evolutionary track and allow graphical representation in a Jupyter Notebook.

Arguments:

name : string
Name of the model.

Keyword arguments:

reinit : boolean
If True, deletes all calculated files.

read : boolean
If True, reads files.HR and.osc

obs : boolean
If True, compute observable quantities such as magnitude and colors.

freq : boolean
If True, GUI is preset with computation of frequency and no computation of model.

create : boolean
If True, the don file is created but Cesam2k20 is not ran.

job : dict
Some options we want to enforce.

CModelGUI.__str__():

Gives the string representation of a CModelGUI instance.

Returns: string Mass, X0, Y0 and Z0 of the model if has been correctly calculated. Otherwise, only mass.

CModelGUI.__call__(mkdon=True, debug=False, edit=True, log=False, devnull=False, **kwargs):

Calculates models or frequencies.

Keyword arguments:

mkdon : boolean, optional
Creates (or re-creates) a.don file if `mkdon==True` ; uses existing.don file otherwise. Default: True.

debug : boolean, optional
If True, run the debug Cesam2k20 executable (`cesam2k20_dbg.x`). Default: False.

edit : boolean, optional
If True, allow to edit.don file or oscillation parameter file.

log : boolean, optional
If True, direct standard output to.log file. Default: False.

devnull : boolean
If True, redirect standard output to /dev/null.

valgrind : boolean, optional
If True, calls Cesam2k20 with valgrind analysis software.

coverage : boolean, optional
If True, evaluate test coverage by calling `cesam2k20_cov.x` with gcov.

CModelGUI.__init():

Method used to initialize all subclasses used by CModelGUI and read files if neaded.

CModelGUI.init_fparamsNB(p_or_g, keep=False):

Initializes parameters for frequency computations. See FreqsGUI for documentation.

Arguments:

p_or_g : string
Chooses whether to calculate p or g modes.

Keyword arguments:

keep : boolean
If True, keeps some options if `CModelGUI.init_fparamsNB` called again.

CModelGUI.hr_axis(ax=None, nologx=False, nology=False, **kwargs):

Sets the axes for plotting tracks on the HRD.

Keyword arguments:

ax : matplotlib.axes._subplots.AxesSubplot
axis instance, optional

nologx : boolean
True if the linear value instead of the log value of the x quantity must be plotted. Default: False.

nology : boolean
True if the linear value instead of the log value of the y quantity must be plotted. Default: False.

**kwargs:

fontsize : float
Font size used for witting axe labels. Default value: `plt.rcParams['font.size']`.

show_ax_label : boolean
Wether we display the ax labels or not. Default: True.

CModelGUI.kiel_axis(ax=None, nologx=False, nology=False, **kwargs):

Sets the axes for plotting tracks on the Kiel Diagram.

Keyword arguments:

ax : ‘matplotlib.axes._subplots.AxesSubplot‘
axis instance, optional

nologx : boolean
True if the linear value instead of the log value of the x quantity must be plotted. Default: False.

nology : boolean
True if the linear value instead of the log value of the y quantity must be plotted. Default: False.

**kwargs:

fontsize : float
Font size used for witting axe labels. Default value: `plt.rcParams['font.size']`.

show_ax_label : boolean
Wether we display the ax labels or not. Default: True.

CModelGUI.hr_box(t, dt, l, dl, logt=True, logl=True, ls='solid'):

Draws an error box on the HRD.

Arguments:

t : float
Effective temperature (or log Teff if `logt=True`)

d :
Error in Teff (or in log Teff if `logt=True`)Error in L (or in log L if `logl=True`)

dt : float

l : float
Luminosity (or log L if `logl=True`)

dl : float

Keyword arguments:

logt : bool, optional
If True, temperature in log (default)

logl : bool, optional
If True, luminosity in log (default)

ls : string
Linestyle to use in plot. Options: `'solid'`, `'dashed'`, `'dashdot'`, `'dotted'`. Default: `'solid'`.

CModelGUI.plot_hr(ls='-', mods=None, ax=None, nologx=False, nology=False, update=False, **kwargs):

Plot Hertsprung-Russel diagramm.

Keyword arguments:

ls : string
Matplotlib supported line style. Default: `'-'`.

mods : list
Draw logteff, logl of specific times steps. Default: None

ax : matplotlib.axes.axes
Instance of matplotlib.axes.axes. Default: Current axis

nologx : boolean
True if the linear value instead of the log value of the x quantity must be plotted. Default: False.

nology : boolean
True if the linear value instead of the log value of the y quantity must be plotted. Default: False.

update : boolean
True if the user wants to update an existing HR diagram. Default: False.

`**kwargs`:

label : str
Label ateached to the evolutionnary track. You need to call `plt.legend()` to display it. Default: None

linewidth : float
Set width of line. Default: Use the one currently set.

imin : int
Index of first time step to be plotted. Defaut: 0

imax : int
Index of last time step to be plotted. Defaut: number of total time step in the model

color : str
Color of the line. Default: Next color in the cycle.

highlight_mods : array of int
Time steps highlighted by a dot

show_ax_label : boolean
Wether we display the ax labels or not. Default: True.

markersize : int
Marker size if marker are used

CModelGUI.plot_kiel(ls=None, mods=None, ax=None, nologx=False, nology=False, update=False, **kwargs):

Plot Kiel diagramm (Teff-logg).

Keyword arguments:

ls : str
Matplotlib supported line style. Default: None.

mods : list
Draw logteff, logg of specific times steps. Default: None

ax : matplotlib.axes.axes
Instance of matplotlib.axes.axes. Default: None

nologx : boolean
True if the linear value instead of the log value of the x quantity must be plotted. Default: False.

nology : boolean
True if the linear value instead of the log value of the y quantity must be plotted. Default: False.

update : boolean
True if the user wants to update an existing Kiel diagram. Default: False.

**kwargs:

label : str
Give label to curve. Default: ”

linewidth : float
Line width of tracks. Default: 2.

imin : int
index of first model in track. Default: 0.

iax : int
index of last model in track. Default: -1.

color : str
Color of the track. Default: next color in color cycle.

CModelGUI.plot_cmd(ls='-'):

Plots a color-magnitude diagram. If B-V and M_V have not been previously computed for model, computes then by calling `self.read_hr(obs=True)`

Keyword arguments:

ls : str
Line style. Default: ’-’.

CModelGUI.plot_cz(r_or_m='m', hatch=False, ax=None, colors=['k'], **kwargs):

Plots a Kippenhahn Diagram of the model's convection zones.

Keyword arguments:

r_or_m :
If 'r', eulerian plot If 'm', Lagragean plot. Default: 'm'.

obs : str

hatch : boolean
Whether or not we draw hatches on different zones. Default: False.

ax : matplotlib.axes.axes
Instance of matplotlib.axes.axes. Default: Current axis

colors : list or numpy.ndarray
List of colors to be cycled through when plotting zones limits. Default: ['k']

Errors raised:

ValueError : r_or_m is not 'm' or 'r'.

CModelGUI.plot_burn(r_or_m='m', hatch=False, color=True, contours=False, with_cz=True, fig=None):

Plots a Kippenhahn Diagram of the model's convection zones.

Keyword arguments:

r_or_m :
If 'r', eulerian plot If 'm', Lagragean plot. Default: 'm'.

obs : str

hatch : booleanbooleanbooleanboolean
Whether or not we draw hatches on different zones. Default: False.

color :
Whether or not we use a color scale (reds). Default: True.

contours :
Whether or not we draw contours. Default: False.

with_cz :
Whether or not we draw convection zones (hatched). Default: True.

Errors raised:

ValueError : r_or_m is not 'm' or 'r'.

CModelGUI.plot_central(limits=False, phases=False):

Plots Tc as a function of rhoc.

Arguments:

limits : boolbool
Sets whether to plot limits of degenerate zones, radiation pressure, etc.

phases :
Sets whether to show evolutionary phase.

CModelGUI.plot_osc(ix, iy, ls='-'):

Plot a column of the.osc file as a function of another one.

Arguments:

ix : int
Index of column represented in abscissa.

iy : int
Index of column represented in ordinates.

Keyword arguments:

ls : str
Line style. Default: '-'.

CModelGUI.plot_all_eps(plot_log=False, age_lims=None, npnt=1000, ncontours=500, zmax=None, zmin=None, cz=False):

no description provided

CModelGUI.plot_all_osc(ivar, plot_log=False, age_lims=None, npnt=1000, ncontours=500, zmax=None, zmin=None, cz=False):

no description provided

CModelGUI.plot_all_osc_r(ivar, plot_log=False, age_lims=None, npnt=2000, ncontours=500, zmax=None, zmin=None, cz=False, rmax=None, fill=True, label=False):

no description provided

CModelGUI.edit_params():

no description provided

CModelGUI.calc_freqsGUI(mods=None, edit=True):

no description provided

CModelGUI.redistrbGUI(amdl, gui=True):

no description provided

CModelGUI.remesh(rep=None, don=None):

no description provided

CModelGUI.plot_ulines(x, xstep=10, ny=50, mag=False, nc=20, colors=None, smooth=False, arrows=False, u_csi=None, quadrant=False):

no description provided

CModelGUI.plot_uvec(x, xstep=10, ny=50, mag=False, normalize=False, smooth=False, maxlength=None, key=None, qx=0.5, qy=0.93, u=None, v=None, quadrant=False):

no description provided

`CModelGUI.streamplot_u(x, smooth=False, density=5, color=None, thickness=None, u=None, v=None, npt=500):`

no description provided

`CModelGUI.plot_echelle(lstyles='', dnu=None, interactive=False, nu0=0.0, freq_obs=None, l_obs=None, nu0_obs=0.0, legn=True, with_lines=False, freq_errors=None):`

no description provided

`CModelGUI.plot_echelle_p(dper):`

no description provided

`CModelGUI.plot_amde(filename=None):`

no description provided

10.2 freqs.py

10.2.1 class FreqsGUI

10.3 cparams.py

10.3.1 class CParametersGUI

`CParametersGUI.__init__(name):`

Class that represents the input parameters of a Cesam2k20 model.

Arguments:

name : string
Name of the model.

10.4 visu_struc.py

10.4.1 class PlotStruc

`PlotStruc.__init__(mod, N2=False):`

no description provided

`PlotStruc.set_xlims():`

no description provided

`PlotStruc.set_ylims():`

no description provided

`PlotStruc.format_ticks(ax):`

no description provided

PlotStruc._xn_changed():

no description provided

PlotStruc._yn_changed():

no description provided

PlotStruc._x_log_changed(new):

no description provided

PlotStruc._y_log_changed(new):

no description provided

PlotStruc._x_inv_changed():

no description provided

PlotStruc._y_inv_changed():

no description provided

PlotStruc._imod_changed():

no description provided

PlotStruc.update():

no description provided

PlotStruc.derivative(x, y):

no description provided

10.4.2 def plot_struc

10.5 visualise.py

10.5.1 class MPLHandler

MPLHandler.close(info, is_ok):

no description provided

10.5.2 class GetNumModel

GetNumModel.__init__(mdl, ax):

no description provided

GetNumModel.onpick(event):

no description provided

GetNumModel.connect():

no description provided

10.5.3 class PlotOscGUI

PlotOscGUI.__init__(models):

Controls the GUI that displays structure of models.

Arguments:

models :
list of Cesam2k20 models.

New attributes:

xn : str
Name of the variable on x-axis.

yn : str
Name of the variable on y-axis.

x_log : bool
True if x-axis is in log scale.

x_inv : bool
True if x-axis is inverted.

y_log : bool
True if y-axis is in log scale.

y_inv : bool
True if y-axis is inverted.

PlotOscGUI.get_varnames():

Retrieves the unicode name of variables avalable in the structure file.

Errors raised:

PYCESAMGUIError : If all osc files are not in the same format ('nadia', 'adia', etc.)

PlotOscGUI.build_dictionnary():

From the variables available for each model, builds a dictionnary with mode names and unicode name of each variables as key.

PlotOscGUI.set_nmod():

Get maximum number of vailable structures for all models.

PlotOscGUI.__get_N2(model, i):

Computes Brunt-Vaissala frequency.

Arguments:

model :
Cesam2k20 model.

i : int
Index of the model in the list.

Returns: list of np.ndarray Brunt-Vaissala frequency.

PlotOscGUI.__get_rR(model, i):

Computes r / Rstar.

Arguments:

model :
Cesam2k20 model.

i : int
Index of the model in the list.

Returns: list of np.ndarray r/Rstar.

PlotOscGUI.__get_rRsun(model, i):

Computes r/Rsun.

Arguments:

model :
Cesam2k20 model.

i : int
Index of the model in the list.

Returns: list of np.ndarray r/Rsun.

PlotOscGUI.__get_mMsun(model, i):

Computes m/Msun.

Arguments:

model :
Cesam2k20 model.

i : int
Index of the model in the list.

Returns: list of np.ndarray m/Msun.

PlotOscGUI.__set_new_var(func, rname, uname):

Defines a new variable visible through the GUI.

Arguments:

func : function
Function that will be applied to compute the new quantity. Should be of the form func(model, i), with model a CModelGUI instance and i an integer.

rname : raw-string
Latex name of the variable.

uname : unicode-string
Unicode name of the variable.

PlotOscGUI.set_composite_vars():

Defines all the new varaibles you need.

PlotOscGUI.initial_plot():

Creates the initial plots with default settings.

PlotOscGUI._xn_changed():

Changes variale on x-axis. Called whenever `self.xn` is changed.

PlotOscGUI._yn_changed():

Changes variable on y-axis. Called whenever `self.yn` is changed.

PlotOscGUI._x_log_changed():

Changes, for x-axis, log scale to linear and vice-versa. Called whenever `self.x_log` is changed.

PlotOscGUI._y_log_changed():

Changes, for y-axis, log scale to linear and vice-versa. Called whenever `self.y_log` is changed.

PlotOscGUI._x_inv_changed():

Inverses x-axis. Called whenever `self.x_inv` is changed.

PlotOscGUI._y_inv_changed():

Inverses y-axis. Called whenever `self.y_inv` is changed.

PlotOscGUI._switch_fired():

Switches variables on x and y axes. Called whenever `self.switch` is changed.

PlotOscGUI._tage_changed():

Finds index of a model's time step with age closest to a target age. Then redraws figure.

PlotOscGUI._txc_changed():

Finds index of a model's time step with core hydrogen to initial core hydrogen ratio closest to a target ratio. Then redraws figure.

PlotOscGUI._trhoc_changed():

Finds index of a model's time step with core density closest to a target core density. Then redraws figure.

PlotOscGUI._tTc_changed():

Find index of a model's time step with core temperature closest to a target core temperature. Then redraws figure.

PlotOscGUI._imod_changed():

Redraws figure with structure corresponding to desired `self.imod`.

PlotOscGUI.__update_age(i):

Gets new value of model's age every time imods changes.

Arguments:

i : int
Index of the mdoel.

PlotOscGUI.__redraw():

Redraws the figure with new settings.

PlotOscGUI.update(i, imod):

Update line data of given model.

Arguments:

i : int
Index of the model in the list.

imod :
Index of the time step whose values should be displayed.int

10.5.4 def polar2cartesian

10.5.5 def movie

10.5.6 def lp2cos

10.5.7 def dlp2cos

10.5.8 def plotpolar

10.5.9 def get_num_model

10.5.10 def set_plot_page

10.6 mpl.py

10.6.1 class _MPLFigureEditor

_MPLFigureEditor.init(parent):

no description provided

_MPLFigureEditor.update_editor():

no description provided

_MPLFigureEditor._create_canvas(parent):

no description provided

10.6.2 class MPLFigureEditor

10.6.3 class MPLInitHandler

MPLInitHandler.init(info):

This method gets called after the controls have all been created but before they are displayed.

10.6.4 class MPLHandler

MPLHandler.close(info, is_ok):

no description provided

10.7 cesam_run.py

10.7.1 class CRunGUI

Chapter 11

Running tests and grids

Part IV

Interface between CESTAM and other codes

Chapter 12

Stellar oscillation programs

Contents

12.1 ADIPLS

12.2 ACOR

Chapter 13

Optimization programs

Contents

13.1 OSM

Contents

13.1.1 osm.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

13.1.2 __init__.py

13.1.3 OSM.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

```
class OSM
```

```
OSM.__init__(parser):
```

no description provided

```
OSM.osm_init(resume=False):
```

no description provided

```
OSM.osm_run(resume=False):
```

no description provided

```
OSM.__process_options():
```

no description provided

```
OSM.write_header(subject):
```

no description provided

```
OSM.write_part_email(ext):
```

no description provided

```
OSM.write_email():
```

no description provided

```
OSM.send_email():
```

no description provided

```
OSM.copy_files():
```

no description provided

```
OSM.get_param_resume(parameters):
```

no description provided

OSM.init_pms_model(model, mass, ftype):

no description provided

OSM.init_zams_model(model, mass, ftype):

no description provided

OSM.create_initial_model():

no description provided

OSM.get_teff():

no description provided

OSM.find_age_tstop():

no description provided

13.1.4 CTarget.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class Target

Target.__init__(xml_elem):

Class reprensenting a targeted value.

Arguments:

xml_elem : DOM Element
XML DOM Element

Target.__repr__():

Defines the representation of a Target.

Returns: str This method outputs the value of <Target.__str__>

Target.__str__():

Return a string describing the target as "name = value +/- sigma".

Returns: str This method outputs the target as "name = value +/- sigma"

Target.copy():

Performs a copy of the <osm3.CTarget> instance.

Returns: '<osm3.CTarget>' A new instance of CTarget with identical attributes.

Target.read_target():

Extract all information relative to a given target, in a XML domain.

13.1.5 CSeismic.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class CSeismic

CSeismic.__init__(modes, lfirst=False):

no description provided

CSeismic.process_data():

no description provided

CSeismic.set_data(data):

no description provided

CSeismic.apply_func(func, param=None):

no description provided

CSeismic.get_constraints(types, lval='file'):

no description provided

CSeismic.get_l_values(modes):

Return l values.

Arguments:

modes : 2d-array
Mode characteristics (n, l, freq, dfreq).

CSeismic.nuu(*args, **kwargs):

no description provided

CSeismic.dnu(*args, **kwargs):

Computes the large separation $\Delta_\nu(n) = nu_{n,l} - nu_{n-1,l}$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : $Delta_nu$ [in muHz] as a function of frequency

CSeismic.d01(kwargs):**

$d01 = nu_{n,0} - \frac{nu_{n-1,1}+nu_{n,1}}{2}$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : d01 [in muHz] as a function of frequency

CSeismic.d02(kwargs):**

$d02 = nu_{n,0} - nu_{n-1,2}$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : d02 [in muHz] as a function of frequency

CSeismic.sd(lval=None):

second difference as a function of frequency (Gough 1990, Houdek & Gough 2007) $sd = nu_{n-1,l} - 2nu_{n,l} + nu_{n+1,l}$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : second difference [in muHz] as a function of frequency

CSeismic.sd01(kwargs):**

sd01 second difference 01 as a function of frequency as defined in Eq. (4) of Roxburgh & Vorontsov (2003,A&A) $sd01(n) = \frac{1}{8}(nu_{n-1,0} - 4nu_{n-1,1} + 6nu_{n,0} - 4nu_{n,1} + nu_{n+1,0})$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : sd01 second difference [in muHz]

CSeismic.sd10(kwargs):**

sd10 second difference "10" as a function of frequency as defined in Eq. (5) of Roxburgh & Vorontsov (2003,A&A) $sd10(n) = -\frac{1}{8}(nu_{n-1,1} - 4nu_{n,0} + 6nu_{n,1} - 4nu_{n+1,0} + nu_{n+1,1})$ INPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency muHz modes[:,3] : sigma error in frequency mHz OUTPUTS: (y,coef): y : sd10 second difference [in muHz]

CSeismic.rsd01(kwargs):**

ratio $sd01(n)/dnu1(n)$ as defined in Eq. 1 of Lebreton & Goupil (2012, A&A) with $dnu1 = nu_{n,1} - nu_{n-1,1}$

CSeismic.rsd10(kwargs):**

ratio $sd10(n)/dnu0(n)$ as defined in Eq. 1 of Lebreton & Goupil (2012, A&A) with $dnu0 = nu_{n+1,0} - nu_{n,0}$

CSeismic.rd02(kwargs):**

ratio $d02(n)/dnu1(n)$ with $dnu1 = nu_{n,1} - nu_{n-1,1}$

13.1.6 CParameter.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

```
class Parameter
```

```
Parameter.__init__(xml_elem=None, p_list=None):
```

Defines the Parameter class. It stores information concenring a parameter given in the input XML file.

Keyword arguments:

xml_elem :
Instance of xml.dom.minidom.Element, read in the input XML file.

p_list : list
List that contains information needed to defin Parameter

New attributes:

name : str
Name of the parameter.

value : any
Value of the parameter

step : float
Step used to vary this parameter in the Levenberg-Marquardt algorithm.

rate : float
Rate of modification of this parameter between two iterations of the Levenberg-Marquardt algorithm.

bounds : list of float
Bound that limit the possibe values of the parameter.

sigma : float
Standard deviation of this parameter. Defined for symmetry with the Target class. It is always set to -1 here

evol : bool
True if the parameter control the evolution

seismic : bool
True if the parameter control the modes

is_input : bool
This attribute is here to keep a symmetry with the CTarget class. It should always be False.

```
Parameter.__repr__():
```

Defines the representation of a Parameter.

Returns: str This method outputs the value of `<Parameter.__str__>`

Parameter.__str__():

Return a string describing the Parameter as "name = value".

Returns: str This method outputs the Parameter as "name = value"

Parameter.__check_compatibility():

Chekcs that the initial value set in the input XML file is inside the specified bounds.

Parameter.copy():

Performs a copy of the <osm3.CParameter> instance.

Returns: A new instance of CParameter with identical attributes.

Parameter.read_parameter():

Extract all information relative to a given parameter, in a XML domain.

13.1.7 CGuess.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

```
class CGuess
```

CGuess.__init__(path=''):

no description provided

CGuess.search_guess():

no description provided

CGuess.search_model_in_grid(old=False):

no description provided

CGuess.files_search(path='./', pattern=''):

no description provided

CGuess.loadhrgrid(files, ntmax=10000, old=False):

files : names of the files (full path) return a tuple (data,masses,ntimes) where data is a Numpy array data[k,j,i]: k : track index (track associated with files[k]) j : index associated with the following quantities: 0 : age (My) 1 : log Teff 2 : log L/Lo 3 : log R/Ro 4 : M/Mo 5 : log g 6 : Xc 7 : Yc 8 : Zc 9 : Xs 10 : Ys 11 : Zs i : time step index (age) Note that age[k] = -1 for i>ntimes[k] masses[k] : masse associated with track k the number of time steps associated with track k

CGuess.search_model(hrin):

no description provided

CGuess.get_min():

no description provided

CGuess.choose_evol_status():

no description provided

CGuess.selected_models(hr):

no description provided

CGuess.get_chi_dist(selected, hr):

no description provided

CGuess.plot_selected_tracks(selected, hr):

no description provided

13.1.8 CSystem.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class CSystem

CSystem.__init__(setup, name, refmodel, verbose=False, model_names=None, refparameter=None):

This class represents a multipls system of stars. It can be a system with only one star in it. Each stellar model in the system is build following the parameters of a reference model. Some modifications to each model can then be set.

Arguments:

name : string
Generic name of the stellar models. Stars of the system are name by the concatenation of the generic name and the star name.

refmodel :
Instance of CModelOSM that represents the reference stellar model.

Keyword arguments:

verbose : bool
If True, prints out details of the run. Default: False.

model_names : list of str
List with the name of stellar models in the system. Default None,

refparameter :
If this instance of CSystem is used to compute the derivative when a given parameter is changed, give an instance of such parameter.

CSystem.__repr__():

no description provided

CSystem.__str__():

no description provided

CSystem.create_stars():

Create the instances of CModelOSM that represent stars in the system. Also fix some parameters that are given for each stars.

CSystem.models_initialization(start, cf, func_pms, func_zams):

Computes an initial model for each star in the system.

Arguments:

start : string

cf : float

Errors raised:

OSMError : osm3 was unable to compute the initial model.

CSystem.set_running(run_all=True):

Goes through all instance of CModelOSM in the system and checks if the models should indeed be computed. If a model is not to be computed, it is decalred as a zombi.

Keyword arguments:

run_all : bool
If True, all models will be computed, i.e., CModelOSM.zombi = False for all models. Default: True.

Errors raised:

OSMError : If run_all is False, the function then checks which models should be computed by lookig at the parameter than should vary for this instance of CSystem. If this name is not given defined through the attribute self.refparameter, an error is raised.

CSystem.cp_pms_files(originals):

no description provided

CSystem.set_attr(sub_attr, attr_name, attr_val):

Changes the value of an attribute, or the attribute of an attribute, of values in `self.stars`.

Arguments:

sub_attr : string
If you want to change the value of the attribute of an attribute, you can provide the name of the first attribute

attr_name : string
Name of the attribute you want to change.

attr_val : any.
Value of the attribute you want to change

CSystem.run_func(sub_attr, func, *args, **kwargs):

Call a given method of the values, or of the attrbute of the values, in `self.stars`.

Arguments:

sub_attr : string
If you want to call the mathod of an attribute, you can provide the name of this attribute

func :
Name of the method to be called.

attr_name : string

attr_val : any.
Value of the attribute you want to change

13.1.9 CSetup.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class CSetup

CSetup.__init__(xml=None):

no description provided

CSetup.__get_nys():

Counts the number of targets associated to each stars (stored in dictionnary `self.nyd`) and count total number for targets, for all stars (stored in `self.ny_tot`).

CSetup.__get_npars():

Counts the number of parameters defined in the XML file (`self.npar`), and the total number of parameter associated to each stars (`self.npar_tot`).

CSetup.__get_ns():

Counts the total number of seismic constraints defined in the XML file (`self.ns_tot`). Also set the indices of seismic constraints in `self.y`, for each star.

CSetup.__get_nt():

Counts the number of targets defined in the XML file (`self.nt`), except for targets defined as inputs.

CSetup.__get_model_settings(xmlmodel):

From a XML domain, extract all information relative to the initialization of Cesam2k20 models.

Arguments:

`xmlmodel` :
XML domain that describes initializations options.

Returns: dict dictionnary of options.

CSetup.__get_levmar_settings(xmllevmar):

From a XML domain, extract all information relative to the Levenberg-Marquardt options.

Arguments:

`xmllevmar` :
XML domain that describes Levenberg-Marquardt options.

Returns: dict dictionnary of options.

CSetup.__get_mcmc_settings(xmlmcmc):

From a XML domain, extract all information relative to the MCMC options.

Arguments:

`xmlmcmc` :
XML domain that describes MCMC options.

Returns: dict dictionnary of options.

CSetup.__get_modes_settings(xmlmodes):

From a XML domain, extract all information relative to the mode computation.

Arguments:

`xmlmodes` :
XML domain that describes mode computation.

Returns: dict dictionnary of options.

CSetup.__get_surface_effects_settings(xmlsurface_effects):

From a XML domain, extract all information relative to the correction of surface effects.

Arguments:

xmlsurface_effects :
XML domain that describes correction of surface effects.

Returns: dict dictionnary of options.

CSetup.__get_common(struct):

From a list of parameters or targets, retrieve instances that are common to all stars.

Arguments:

struct : list
A list of parameters or targets.

Returns: (list, int, <numpy.ndarray>) Returns a list of common elements, a flag which is true if the XML describes a multiple system, an array of star names.

CSetup.__read_seismic_constraints(xmlseismic_constraints):

From a XML domain, extract all information relative to the seismic constraints.

Arguments:

xmlseismic_constraints :
XML domain that describes the seismic constraints.

Returns: dict dictionnary of options.

CSetup.copy():

Performs a copy of the <osm3.CSetup> instance.

Returns: A new instance of CSetup with identical attributes.

CSetup.read_setup():

Read input XMl file and extract setup options.

13.1.10 CModelOSM.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class CModelOSM

CModelOSM.__init__(name, star='ref', reinit=False, read=True, verbose=True, index=0):

no description provided

CModelOSM.__get_chem_abund_diff(Params):

no description provided

CModelOSM.__get_enrichment(y0, zsx0):

no description provided

CModelOSM.setup_don_file(setup=None, parameters=None, settings=None):

no description provided

CModelOSM.clean_files():

no description provided

CModelOSM.get_output(arg):

no description provided

CModelOSM.f_3D_out(arg):

no description provided

CModelOSM.f_omega_s(arg):

no description provided

CModelOSM.f_feh_s(arg):

no description provided

CModelOSM.f_li_s(arg):

no description provided

CModelOSM.f_tczst0(arg):

Get ratio of CZ accoustic radius to total accoustic radius.

CModelOSM.f_tcz(arg):

Get CZ accoustic radius.

CModelOSM.f_r_cz(arg):

no description provided

CModelOSM.set_run_options(job=None, type_file=None, mod_init=None, c_iben=None):

no description provided

13.1.11 CComputeOptimal.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

```
class ComputeOptimal
```

```
ComputeOptimal.__init__(generic_name, model_names, setup, Aux3DData=None, verbose=False, debug
=False):
```

This class lead the computation of an optimal model corresponding best to a set of constraints. This optimal model can be found either using a Levenberg-Marquardt, or a MCMC agorithm (not yet functional).

Arguments:

generic_name : str
Generic name of the OSM run.

model_names : list of str
List of the model names for each stars.

setup :
OSM options stored in a <osm3.CSetup> instance.

Keyword arguments:

Aux3DData : dict
Data taken from a 3D RHD model. Needed to build a patched model. Default: None.

verbose : bool
If True, print additional information to standard output. Default: False.

debug : bool
If True, run Cesam2k20 in debug mode. Default: False.

```
ComputeOptimal.__get_seismic_outputs(model, new_parameters):
```

Fetch seismic properties of the models, in the correct range of frequencies.

Arguments:

model :
The model for which modes must be computed.

new_parameters :
The set of modified parameters

Returns: list of float Returns a list of observable corresponding to the quantities given as constraints.

ComputeOptimal.__get_outputs(model, new_parameters):

Fetch properties of the model, corresponding to targeted observables (except seismic constraints).

Arguments:

model :
The model for which modes must be computed.

new_parameters :
The set of modified parameters

Returns: list of float Raises: OSMError: Description Returns a list of observable corresponding to the quantities given as constraints.

ComputeOptimal.log_likelihood(model, params):

Computes the log of likelihood.

Arguments:

model :
The model for which the log likelihood should be computed.

params :
The set of parameters

Returns: float The log likelihood.

ComputeOptimal.log_prior(params):

Computes the log prior.

Arguments:

params :
The set of parameters

Returns: float The log of prior.

ComputeOptimal.log_probability(model, params):

Computes the log of the probability.

Arguments:

model :
The model for which the log probability should be computed.

params :
The set of parameters

Returns: float The log probability.

ComputeOptimal.compute_model_MCMC(vals):

Compute one model for the MCMC algorithm. It set properly the parameters of the models, its initial conditions and returns the log probability that this model is the optimal one.

Arguments:

vals :
Values given to each parameter
list : of float

Returns: float the log probability that this model is the optimal one.

Errors raised:

OSMError : If the.amdl file could not be produced.

ComputeOptimal.compute_model_LM(parameters, args, status):

Compute one model for the Levenberg-Marquardt algorithm: it is the to be function evaluated. It set properly the parameters of the models, its initial conditions and returns the log probability that this model is the optimal one.

Arguments:

parameters : List of ‘<osm3.CParameter>‘
Parameters used for the model to be computed.
args : (‘<osm3.CModelOSM>‘, ‘<osm3.CModelOSM>‘, list of CTarget)
Additional arguments passed to the function. Usually args is the model, the associated central model and the set of targeted parameters.
status : int
Index that corresponds to the parameter tested. If `status = -1`, it corresponds to the central. Otherwise, this model is computed in order to evaluate derivate with respect to `parameter[status]`.

Returns: list of float The values of target observables.

Errors raised:

OSMError : If the.amdl file could not be produced.

ComputeOptimal.init_models_LM():

Initialise the data structures used to run the Levenberg-Marquardt algorithm. We create a dictionnary with the following structure: `{'center':CSystem, 'der': {'param1': CSystem,..., 'paramN': CSystem}}`. `'center'` correpsonds to the central system, with curent optimal parameters. `'der'` is the directory of all systems, used to compute derivatives, with respect to `param1`,..., `paramN`.

ComputeOptimal.init_models_MCMC():

Creates initial reference model for MCMC algorithm.

ComputeOptimal.print_init_params():

Prints to standard output the initial choice of parameters, and the targeted quantities.

ComputeOptimal.init_stats():

Creates array with aggregated non-seismic and seismic targets, covariance matrix, etc. Also stores in memory parameters that control some criterion of the Levenberg-Marquardt algorithm.

ComputeOptimal.write_flush(text):

Prints message to standard output, write the same message in the log file, and flush buffer.

Arguments:

text : str
Message to be written.

ComputeOptimal.call_levmar():

Wrapper that call the CLevMar class, through its `__call__` method.

Returns: str Returns the text that will be written in the log file.

Errors raised:

OSMError : If some error occured in `<osm3.CLevMar.__call__>`.

ComputeOptimal.call_mcmc():

Wrapper that call the CLevMar class, through its `run_mcmc` method.

ComputeOptimal.clean_models():

Remove temporary models.

ComputeOptimal.print_final_results(msg):

Print the final results of OSM run.

Arguments:

msg : str
Messgae to be written.

13.1.12 CLevMar.py

class LevMar

LevMar.__init__(setup, model_names, y, sigma, covar, yname, f):

Class that implements the Levenberg-Marquardt algorithm. (See Numerical Recipes, Press et al. for details.) χ^2 is a merit function that can be defined as a distance between targeted observable, and optimal observable values. Close to global optimal: $\chi^2 \simeq 0 \simeq \gamma - d \cdot a - 0.5a \cdot D \cdot a$ a_l are the current choice of parameters, d and D are respectively 1st and 2nd derivative of the χ^2 We denote $\beta_k = -0.5\partial\chi^2/\partial a_k$ and $\alpha_{kl} = 0.5\partial^2\chi^2/\partial a_k/\partial a_l$ Correction to optimal choice of parameters are then computed with: $\sum_{l=0}^{M-1} \alpha_{kl}\delta a_l = \beta_k$.

Arguments:

setup : '<osm3.CSetup>'
Setup option read in input XML file.

model_names : List of str
Names of the Cesam2k20 models representing each star.

y : 1d '<numpy.array>'
Target values.

sigma : 1d '<numpy.array>'
Standard deviation for classical (i.e. not seismic) constraints

covar : 2d '<numpy.array>'
Covariance matrix. `covar(i,i) = target_i.sigma**2`.

f : '<class '_io.TextIOWrapper'>'
Text stream.

LevMar.__call__(func, levmar_args, verbose=True, ftol=1e-3, maxiter=30, chi2min=1e-4, lamb0=1e-4, cov_cdtnb_thr=1e13, hess_cdtnb_thr=1e13):

Launch the Levenberg-Marquardt algorithm.

Arguments:

func : function
`f_unc(par,args,status)` is the function that computes the model and that we want to search for the optimal parameters. See `<osm3.CComputeOptimal.compute_model_LM>` for a complete documentation of this function.

levmar_args : (dict of '<osm3.CSystem>', list of 'CTarget')
Arguments of the function that computes a model for the Levenberg-Marquardt algorithm.

Keyword arguments:

verbose : bool
True if additional informtion is to be printed. Default: True.

ftol : float
Minimum variation of chi2 allowed. Below this threshold, we consider that the optimization has reach a local minimum. Defaut: 1e-3.

maxiter : int
Maximum number of iterations. Default: 30.

chi2min : float
chi2 threshold below which the solution is accepted. Default: 1e-4.

lamb0 : float
Initial damping parameter. Default: 1e-4

cov_cdtnb_thr : float
adopted threshold for the condition number associated with the co-variance matrix.

hess_cdtnb_thr : float
adopted threshold for the condition number associated with the final Hessian matrix.

Returns: (float, list of <osm3.CParameter>, list of float, int, bool, str) Tuple composed of the best chi2, best parameter, best observables, the number of iterations, an error flag and the text that must be written to file.

Errors raised:

OSMError1 : There is a null eigenvalue in the Hessian matrix, it cannot be inverted

OSMError2 : Unable to compute a model.

LevMar.__add_process(ipar, target_func, parameters, levmar_args):

For each tunable parameter, this function adds the necessary processes to the queue/ These processes are computations of a Cesam2k20 model with given input parameters.

Arguments:

ipar : integer
Corresponds to the index of the parameter (ipar - 1). If ipar = 0, it corresponds to the central, un-shifted model.

target_func : function
The function that should be called to compute the models.

parameters : List of '<osm3.CParameter>'
List of the paremeters defined in the input XML file.

levmar_args : (dict of '<osm3.CSystem>', list of 'CTarget')
Arguments passed to <osm3.CLevMar.__call__>.

Returns: integer Returns the number of new created processes.

LevMar.__reorder_outputs(array_in, array_out, star):

Reorder output values stored in an array, so that each element correspond to the right target of a given star.

Arguments:

array_in : ld '<numpy.ndarray>'
Input data.

array_out : ld '<numpy.ndarray>'
Output data.

star : str
Star whose outputs are given

LevMar.__rename_optimal_model():

Rename the file of optimal model by adding 'fin_' prefix.

LevMar.__get_func_args_tmp(levmar_args, pname, pstar):

Builds the function arguments for `func` or `func_der` from the dictionnary of stellar system, the parameter name and the name of a star.

Arguments:

levmar_args : (dict of '<osm3.CSystem>', list of CTarget)
Tuple containing the dictionnary of CSystem instances and the list of CTarget.

pname : string
Name of the parameter.

pstar : string
Name of the star.

Returns: ('<osm3.CModelOSM>', '<osm3.CModelOSM>', list of CTarget) Tuple of (shifted CModelOSM with given star name; center CModelOSM, list of CTargets

LevMar.func_der(func, parameters, args, index):

Computes a models with a shifted parameter.

Arguments:

func : function
Function designed to compute a Cesam2k20 model and returns the value of targets.

parameters : list of '<osm3.Parameter>'
List of `<osm3.Parameter>` instances defined in input XML file.

args : ('<osm3.CModelOSM>', '<osm3.CModelOSM>', list of CTarget)
Arguments passed to function `func`.

index : int
Index corresponding to the shifted parameter.

Returns: (list of float, float, bool) Values of target for shifted model, vaue of the step and err.

Errors raised:

OSMError : If model computation did not finished.

LevMar.func_shell(func, parameters, args, index, pipe):

Wrapper used to run the function `self.func` in parallel. Args:

Arguments:

func : function
The function to be evaluated.

parameters : list of '<osm3.CParameter>'
List of CParameter instances.

args : ('<osm3.CModelOSM>', '<osm3.CModelOSM>', list of CTarget)
Additional arguments passed to the function. Usually args is the model, the associated central model and the set of targeted parameters.

index : int
index corresponding to the parameter shifted. `index=-1` for central model

pipe : tuple
Pipe that connect two connection objects.

LevMar.func_der_shell(func, parameters, args, index, pipe):

Wrapper used to run the function `self.func` in parallel, to compute derivative with respect to a given parameter. Args:

Arguments:

func : function
The function to be evaluated.

parameters : list of '<osm3.CParameter>'
List of CParameter instances.

args : ('<osm3.CModelOSM>', '<osm3.CModelOSM>', list of CTarget)
Additional arguments passed to the function. Usually args is the model, the associated central model and the set of targeted parameters.

index : int
index corresponding to the parameter shifted. `index=-1` for central model

pipe : tuple
Pipe that connect two connection objects.

LevMar.chi2(y_mod, y, W):

Compute the chi2 value.

Arguments:

y : 1d array of float
Targets' values.

y_mod : 1d array of float
Values of targets from models.

W : 2d array of float
Inverse of the covariance matrix.

Returns: float The chi2 vlue

LevMar.levmar_init(levmar_args):

Initialize some quantities need to run the Levenberg-Marquardt algorithm, and compute the Jacobian matrix for 1st iteration.

Arguments:

levmar_args : (dict of '<osm3.CSystem>', list of CTarget)
Tuple containing the dictionnary of CSystem instances and the list of CTarget.

Returns: str Message that should be written to file

Errors raised:

OSMError1 : There is a null eigenvalue in the covariance matrix.

OSMError2 : Unable to compute 1st model.

LevMar.levmar_der(centre_mod, parameters, levmar_args, jac):

Finds the next optimal model. Executes one step of the Levenberg-Marquardt algorithm.

Arguments:

centre_mod : ‘numpy.ndarray‘
Values of the central models (not shifted).

parameters : list of ‘<osm3.CParameter>‘
List of CParameter instances.

levmar_args : (dict of ‘<osm3.CSystem>‘, list of ‘CTarget‘)
Arguments of the function that computes a model for the Levenberg-Marquardt algorithm.

jac : 2d ‘numpy.ndarray‘
Jacobian of the system: $\partial T_j/\partial P_i = (T_j(P_i + \delta p_i) - T_j(p_i))/(\delta p_i)$.

Returns: (bool, numpy.ndarray, numpy.ndarray) Tuple composed of an error flag, the central values and the jacobian.

Errors raised:

OSMError : l: At least one of the models crashed.

LevMar.levmar_coef(y_mod, npar, der, lamb):

Computes coefficients of the Levenberg-Marquardt algorithm: α_{kl} and β_k . See <CLevMar.__init__> for details

Arguments:

y_mod : 1d ‘<numpy.ndarray>‘
Value of observable, obtained from a model.

npar : int
Number of parameters, size of y_mod

der : 2d ‘<numpy.ndarray>‘
Jocobian matrix of the system.

lamb : float
Damping parameter λ .

Returns: (2d ‘<numpy.ndarray>‘, 1d ‘<numpy.ndarray>‘) α_{kl} and β_k

LevMar.hessian2str(alpha, parameters):

Convert the hessian matrix to an easily readable string.

Arguments:

alpha : 2d ‘<numpy.ndarray>‘
Coefficients of the hessian matrix

parameters : List of ‘<osm3.Parameter>‘
List of Parameters defined in XML file.

Returns: str String representation of the Hessian matrix.

LevMar.print_init():

Print information on the initial state of the run and chosen settings.

LevMar.print_res(best_chi2, new_chi2=None):

Print a summary of the result of each iteration of the algorithm.

Arguments:

best_chi2 : float
Best χ^2 value.

new_chi2 : float
New χ^2 value. Default: None.

Returns: str Text that should be printed out and written to.log file.

13.1.13 utils.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

class OSMError

OSMError.__init__(message):

Define Exception raised for any problem encountered in OSM

Arguments:

message : str
The error message.

OSMError.__str__():

String representation of the class.

Returns: str The string representation of the class is the message written in self.message.

class Aux3DDataMissing

Aux3DDataMissing.__init__():

Define Exception raised for any problem encountered in OSM, specific to patched models.

def write_amdl

def cesam2amdl

def get_inst

def get_first_value

def setattr_l

13.1.14 osmmodes.py

Optimal Stellar Models (OSM) Copyright (c) 2012 R. Samadi (LESIA - Observatoire de Paris) This is a free software: you can redistribute it and/or modify it under the terms of the GNU General

Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this code. If not, see <http://www.gnu.org/licenses/>.

```
class SeismicConstraints
```

```
SeismicConstraints.__init__(file='', types=['nu'], matching='frequency', data=None, lfirst=False,
star=''):
```

no description provided

```
SeismicConstraints.print_freqs():
```

no description provided

```
SeismicConstraints.print_constraints():
```

no description provided

```
class SeismicModel
```

```
SeismicModel.__init__(file, index=0):
```

no description provided

```
SeismicModel.read_mad(file):
```

OUTPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency in muHz (eigenvalue) modes[:,3] : square normalised frequency (w2) modes[:,4] : Richardson frequency in muHz

```
SeismicModel.read_agsm(file):
```

OUTPUTS: modes[:,0] : n order modes[:,1] : l degree modes[:,2] : nu frequency in muHz (eigenvalue) modes[:,3] : square normalised frequency (w2) modes[:,4] : Richardson frequency in muHz modes[:,5] : variationnal frequency in muHz modes[:,6] : icase modes[:,7] : Inertia

```
SeismicModel.__surface_effects_prescription(model, coeff, lsep):
```

no description provided

```
SeismicModel.largesep(model, Params, lsep_target, Settings, se=True):
```

Compute large separation.

Arguments:

Params : list of osm3.Parameter instances.
List of tunable parameters.

lsep_target : Target instance
instance of osm3.Target representing the targeted large sep.

Settings : Dictionnary
Settings of the optimization.

se : boolean
If True, then large separation is computed on surface effect corrected frequencies.

SeismicModel.__get_surface_effects(Params, surface_effects, lsep, model):

Provide correction of the surface effect.

SeismicModel.select_mode_set(modesset, lsep_target):

no description provided

SeismicModel.seismic_model(setup, model):

no description provided

13.2 AIMS

Appendix A

Données diverses

On donne la liste des tableaux glob et var créés par cesam, *cf.* §7.7.3 (Page 158), pour les sorties ASCII, ainsi que celle du tableau list_cesam servant à la compilation et à la création du module exécutable, *cf.* §?? (Page ??).

A.1 Liste du tableau glob

```
glob(1)=mstar*msol
glob(2)=rtot*rsol
glob(3)=ltot*lsol
glob(4)=z0
glob(5)=x0
glob(6)=alpha
glob(7)=9./4.
glob(8)=1./162.
glob(9)=X dans ZC
glob(10)=Y dans ZC
glob(11)=d2p
glob(12)=d2ro
glob(13)=age
glob(14)=wrot vitesse de rotation globale
glob(15)=w_rot initial

nglob=15
```

A.2 Liste du tableau var

```
var(1,i): Rayon
var(2,i): Ln M/Mtot
var(3,i): Température
var(4,i): Pression
var(5,i): Densité
var(6,i): Gradient
var(7,i): Gradient
var(8,i): Luminosité
var(9,i): Opacité
var(10,i): Energie nuc+grav
```

```
var(11,i): Grand Gamma1
var(12,i): Gradient adiabatique
var(13,i): Delta
var(14,i): Cp
var(15,i): Mue^(-1)
var(16,i): Mu
var(17,i): Vaissala
var(18,i): Omega
var(19,i): dln kappa/dln T
var(20,i): dln kappa/dln ro
var(21,i): d epsilon(nuc) / d ln T
var(22,i): d epsilon(nuc) / d ln ro
var(23,i): !Ptot/Pgaz
var(24,i): !Gradient radiatif
var(25,i): d Gamma1 / d lnP (TY)
var(26,i): d Gamma1 / d lnT (PY)
var(27,i): d Gamma1 / dY    (PT)
var(28,i): dP / dro (TX)
var(29,i): dP / dT (roX)
var(30,i): dP / dX (Tro)
var(31,i): du / dro (TX)
var(32,i): du / dT (roX)
var(33,i): du / dX(Tro)
var(34,i): énergie interne
var(35,i): d^2P / dro^2 (TX)
var(36,i): d^2P / dro dT (X)
var(37,i): d^2P / dT^2(roX)
var(38,i): d^2U / dro^2 (TX)
var(39,i): d^2U / dro dT (X)
var(40,i): d^2U / dT^2 (X)
var(41,i): dK / dX
var(42,i): d^2K / dT^2
var(43,i): d epsi / dX
var(44,i): dX / dR
var(45,i): J-B
var(46,i): Edding. facteur
var(ivar+j,i): xchim1g(j=1,nbelem) Abondances / gramme

ivar=46
```

Appendix B

Liste des fichiers binaires

Les Fichiers binaires n'ayant pour fonction que des exploitations internes, leur contenu n'est pas détaillé. A l'exception du fichier binaire d'atmosphère `mon_modele.atm`, ces fichiers peuvent être lus par la routine `lit_binaire`, *cf.* §7.44 (Page 206).

- Fichier binaire du modèle de pré-séquence principale homogène: `mon_modele_B.pms`.
- Fichier binaire du modèle de séquence principale d'âge zéro homogène: `mon_modele_B.hom`.
- Fichier binaire du modèle de séquence principale d'âge zéro: `mon_modele_B.zams`, créé à la fin de ma PMS dès que l'énergie d'origine thermonucléaire devient ègale à l'énergie d'origine graviphique.
- Fichier binaire du modèle de post-séquence principale, créé lorsque l'abondance d'hydrogène au centre devient inférieure à 0.001: `mon_modele_B.post` ou `mon_modele_B.tams`.
- Fichier binaire du modèle du début de la combustion de l'hélium, créé dès que la température au centre dépasse 10^8K: `mon_modele_B.cohe`.
- Fichier binaire du modèle du début de la combustion du carbone, créé dès que la température au centre dépasse $6\,10^8$K: `mon_modele_B.coca`.
- Fichier binaire du modèle du début de la combustion de l'oxygène, créé dès que la température au centre dépasse 10^9K: `mon_modele_B.coox`.
- Fichier binaire du modèle final: `mon_modele_B.dat`.
- Fichier binaire du modèle intermédiaire: `mon_modele_B.rep`.
- Fichier binaire du modèle d'atmosphère: `mon_modele_B.atm`.

Appendix C

Liste des fichiers ASCII

Se reporter §3.3 (Page 14) pour la description du fichier ASCII de données `mon_modele.don`.

C.1 Fichiers pour oscillations

Les fichiers ASCII pour les oscillations ont une en-tête commune. Leur contenu diffère ensuite suivant leur type.

`en-tête` :

- Lignes 1-4 : identification et physique utilisée.
- Ligne 5 : nombre, noms des éléments chimiques utilisés.
- Ligne 6 : nombre de couches, de "global" (13), de variables, d'éléments chimiques (redondant), indice de la vitesse angulaire (s'il n'en est pas tenu compte, l'indice est -1).

Exemple :

```
Fichier pour inversion: test-inv.osc
CESAM2k version 0.0.0.0 lagr colloc  1  2 np no diffus, 31 Aout 2003 17h45
Physique utilisée: etat_eff, opa_int_zsx, conv_jmj, ppcno9, NACRE
solaire_gn, lim_atm, hopf, perte_ext, diffm_mp, difft_nu, ctes_94
 10  H1  He3  He4  C12  C13  N14  N15  O16  O17  Si28
       472        13        25        10        -1
 1.989190000000E+33 6.959888774683E+10 3.846010241481E+33 1.723296157963E-02 7.033715921640E-01
 1.601359931639E+00 7.033715881909E-01 2.793954502293E-01-1.234804262016E+02-1.250212463069E+02
 4.650000000000E+03 0.000000000000E+00 0.000000000000E+00
 6.965125316795E+10 1.125188830934E-10 4.712514552821E+03 1.082443642936E+03 3.550000000000E-09
 2.317104979700E-04 3.846010241481E+33 5.992732100399E-03 0.000000000000E+00 1.664711685203E+00
 3.986098681160E-01 1.004603449555E+00 1.630687926937E+08 1.300571860485E+00 2.500897748491E+03
 0.000000000000E+00 1.729836584721E+00 8.429278614488E-01 0.000000000000E+00 0.000000000000E+00

.........................................

 3.983397251906E-01 9.536638658353E-01 2.457927862174E+08 8.289918269170E-01 0.000000000000E+00
 0.000000000000E+00-1.948967682005E+00 3.878822249046E-01 2.023289474023E+02 1.828866051074E+01
 1.000000000000E+00 3.288637435509E-01-5.586998681649E-03 6.642708700530E-03-8.894901969547E-03
 3.750487629847E-01 1.039826643345E-05 6.071493899575E-01 1.637161572818E-05 4.475726236722E-06
 4.374234929779E-03 1.935496963607E-07 7.968745830613E-03 3.970500883875E-04 5.030377050940E-03
```

C.1.1 Fichier pour oscillations adiabatiques

```
glob: variables globales du fichier mon_modele-ad.osc

glob(1)=mstar*msol
glob(2)=rtot*rsol
glob(3)=ltot*lsol
glob(4)=z0
```

```
glob(5)=x0
glob(6)=alpha
glob(7)=X dans ZC
glob(8)=Y dans ZC
glob(9)=d2p
glob(10)=d2ro
glob(11)=age
glob(12)=wrot vitesse de rotation globale
glob(13)=w_rot initial
glob(14)=g constante de la gravitation utilisée
glob(15)=msol masse solaire utilisée
glob(16)=rsol rayon solaire utilisé
glob(17)=lsol luminosité solaire utilisée

var: variables locales utilisées ; nvar=22 pour oscillations adiabatiques

var(1,i)=r*rsol
var(2,i)=log(m/mstar) -1.d38 au centre
var(3,i)=t
var(4,i)=Ptot
var(5,i)=ro
var(6,i)=gradient réel d ln T / d ln P
var(7,i)=l
var(8,i)=kap
var(9,i)=énergie thermo+gravifique
var(10,i)=grand Gamma1
var(11,i)=gradient adiabatique
var(12,i)=delta
var(13,i)=cp
var(14,i)=mu elec.
var(15,i)=vaissala, 0 au centre
var(16,i)=vitesse angulaire, radian/sec
var(17,i)=d ln kappa / d ln T
var(18,i)=d ln kappa / d ln ro
var(19,i)=d epsilon(nuc) / d ln T
var(20,i)=d epsilon(nuc) / d ln ro
var(21,i)=Ptot / Pgas (grad_mu sans pression turbulente)
var(22,i)=gradient radiatif

composition chimique

var(22+j,i)=xchim(j)*nucleo(j), j=1,nbelem
```

C.1.2 Fichier pour oscillations non adiabatiques

```
glob: variables globales du fichier mon_modele-nad.osc

var: variables
glob(1)=mstar*msol
```

```
glob(2)=rtot*rsol
glob(3)=ltot*lsol
glob(4)=z0
glob(5)=x0
glob(6)=alpha
glob(7)=X dans ZC
glob(8)=Y dans ZC
glob(9)=d2p
glob(10)=d2ro
glob(11)=age
glob(12)=wrot vitesse de rotation globale
glob(13)=w_rot initial
glob(14)=g constante de la gravitation utilisée
glob(15)=msol masse solaire utilisée
glob(16)=rsol rayon solaire utilisé
glob(17)=lsol luminosité solaire utilisée

var: variables locales utilisées ; nvar=44 pour oscillations non adiabatiques

var(1,i)=r*rsol
var(2,i)=log(m/mstar) -1.d38 au centre
var(3,i)=t
var(4,i)=Ptot
var(5,i)=ro
var(6,i)=gradient reel d ln T / d ln P
var(7,i)=l
var(8,i)=kap
var(9,i)=énergie thermo+gravifique
var(10,i)=grand Gamma1
var(11,i)=gradient adiabatique
var(12,i)=delta
var(13,i)=cp
var(14,i)=mu elec.
var(15,i)=vaissala, 0 au centre
var(16,i)=vitesse angulaire, radian/sec
var(17,i)=d ln kappa / d ln T
var(18,i)=d ln kappa / d ln ro
var(19,i)=d epsilon(nuc) / d ln T
var(20,i)=d epsilon(nuc) / d ln ro
var(21,i)=Ptot / Pgas (grad_mu sans pression turbulente)
var(22,i)=gradient radiatif
var(23,i)=d Gamma1 / d log P
var(24,i)=d Gamma1 / d log T
var(25,i)=d Gamma1 / dY = d Gamma1 / dZ
var(26,i)=dP / dro (TX)
var(27,i)=dP / dT (roX)
var(28,i)=dP / dX (Tro)
var(29,i)=du / dro (TX)
var(30,i)=du / dT (roX)
```

```
var(31,i)=du / dx(Tro)
var(32,i)=énergie interne
var(33,i)=d^2P / dro^2 (TX)
var(34,i)=d^2P / dro dT (X)
var(35,i)=d^2P / dT^2(roX)
var(36,i)=d^2U / dro^2 (TX)
var(37,i)=d^2U / dro dT (X)
var(38,i)=d^2U / dT^2 (X)
var(39,i)=dK / dX
var(40,i)=d^2K / dT^2
var(41,i)=d epsi / dX
var(42,i)=dX / dR
var(43,i)=J-B
var(44,i)=Edding. facteur

composition chimique

var(44+j,i)=xchim(j)*nucleo(j), j=1,nbelem
```

C.1.3 Fichier pour inversions

```
glob: variables globales du fichier mon_modele-inv.osc

glob(1)=mstar*msol
glob(2)=rtot*rsol
glob(3)=ltot*lsol
glob(4)=z0
glob(5)=x0
glob(6)=alpha
glob(7)=X dans ZC
glob(8)=Y dans ZC
glob(9)=d2p
glob(10)=d2ro
glob(11)=age
glob(12)=wrot vitesse de rotation globale
glob(13)=w_rot initial
glob(14)=g constante de la gravitation utilisée
glob(15)=msol masse solaire utilisée
glob(16)=rsol rayon solaire utilisé
glob(17)=lsol luminosité solaire utilisée

var: variables locales utilisées ; nvar=25 pour inversion

var(1,i)=r*rsol
var(2,i)=log(m/mstar) -1.d38 au centre
var(3,i)=t
var(4,i)=Ptot
var(5,i)=ro
var(6,i)=gradient reel d ln T / d ln P
var(7,i)=l
```

```
var(8,i)=kap
var(9,i)=énergie thermo+gravifique
var(10,i)=grand Gamma1
var(11,i)=gradient adiabatique
var(12,i)=delta
var(13,i)=cp
var(14,i)=mu elec.
var(15,i)=vaissala, 0 au centre
var(16,i)=vitesse angulaire, radian/sec
var(17,i)=d ln kappa / d ln T
var(18,i)=d ln kappa / d ln ro
var(19,i)=d epsilon(nuc) / d ln T
var(20,i)=d epsilon(nuc) / d ln ro
var(21,i)=Ptot / Pgas (grad_mu sans pression turbulente)
var(22,i)=gradient radiatif
var(23,i)=d Gamma1 / d log P
var(24,i)=d Gamma1 / d log T
var(25,i)=d Gamma1 / dY = d Gamma1 / dZ

composition chimique

var(25+j,i)=xchim(j)*nucleo(j), j=1,nbelem
```

C.1.4 Utilisation du nombre maximum de couches

Avec certains types de précision, *cf.* § 3.3 (Page 14), il est possible de créer un fichier ASCII d'oscillation comportant au moins un nombre de couches fixé à l'avance; le nombre maximum de couches étant utilisé lors du calcul des derniers modèles. Les critères requis pour ce faire sont les suivants :

- Type de précision utilisé: 'sa', ou 'co'.
- ***ET*** réalisation d'un des critères suivants :
  - Age du modèle à calculer inférieur d'un million d'années de l'âge maximum à atteindre.
  - Numéro du modèle à calculer égal à `NMAX_MODELS - 1`.
  - Température effective voisine de la limite demandée.
  - Abondance centrale d'hydrogène voisine de la limite demandée.
  - Extension du noyau d'hélium voisine de la limite demandée.

C.2 Fichier pour diagramme HR

Le fichier pour tracé/exploitation du diagramme HR est systématiquement implémenté au cours de chaque exécution de Cesam2k20. Il est créé lors de l'initialisation d'un modèle sur la séquence principale d'âge zéro homogène ou la pré-séquence principale homogène. Il est complété en cas de reprise d'un modèle existant.

Pour chaque modèle, sur une première ligne sont reportés :

1. l'âge en 10^6 ans;
2. le nombre d'éléments du vecteur de composition chimique;

3. le nombre de limites zones radiatives / zones convectives
4. le numéro du modèle,
5. pour chaque limite RZ/CZ la mention "T" ou "F" suivant qu'il s'agit, ou non, du début d'une zone convective;
6. suivant le nombre de limites, une ou plusieurs lignes sont nécessaires pour indiquer :
   (a) $\log T_{\rm eff}$,
   (b) $\log L/L_\odot$,
   (c) $\log R/R_\odot$
   (d) $M_\star/M_\odot$
   (e) pour chaque limite, $(M_\star - m)/M_\odot$, $r/R_\odot$, $r_{\rm ov}/R_\odot$, ($r_{\rm ov}$ étant le rayon de la limite étendue par overshooting).

Dans les cas particuliers, on utilise les conventions suivantes :

- modèle totalement convectif: 1 (seule) limite "F" placée au centre ($r/R_\odot = 0$),

- modèle totalement radiatif: 0 (aucune) limite "F" placée en -100 ($r/R_\odot = -100$).

Viennent ensuite les noms des éléments chimiques et leurs abondances, par unité de masse, au centre et à la surface. S'il n'est pas imposé d'overshoot, les valeurs $r_{\rm ov}$ des rayons des limites étendues sont fixées à -100. S'il est tenu compte de la diffusion du moment cinétique, une ligne supplémentaire, identifiée pr `Wrot` indique les vitesses linéaire et angulaire de la couche externe. Exemple :

```
 0.000000000000000E+00 10  2   0 F T
 3.749146E+00-1.723498E-01-6.098130E-02 1.000000E+00 9.053737E-01 1.262897E-01
-1.000000E+02 2.969686E-02 6.310311E-01-1.000000E+02
 H1  6.97718E-01 6.97718E-01
He3  8.91487E-05 8.91487E-05
He4  2.82700E-01 2.82700E-01
C12  3.33842E-03 3.33842E-03
C13  4.02357E-05 4.02357E-05
N14  1.03227E-03 1.03227E-03
N15  4.06226E-06 4.06226E-06
O16  9.39680E-03 9.39680E-03
O17  3.80402E-06 3.80402E-06
Si28 5.67649E-03 5.67649E-03
Wrot 5.00050E+01 4.92437E-05  <-- qu'avec diffusion du moment cinétique
 1.000000000000000E+01 10  2   1 F T
 3.750235E+00-1.660591E-01-5.995672E-02 1.000000E+00 9.485921E-01 1.002733E-01
-1.000000E+02 2.900062E-02 6.330032E-01-1.000000E+02
 H1  6.97097E-01 6.97795E-01
He3  6.62006E-05 8.91324E-05
He4  2.83225E-01 2.82627E-01
C12  2.45937E-03 3.33777E-03
C13  4.11117E-04 4.02273E-05
N14  1.66274E-03 1.03208E-03
N15  2.18466E-07 4.06144E-06
O16  9.39741E-03 9.39506E-03
O17  3.87613E-06 3.80329E-06
Si28 5.67687E-03 5.67553E-03
Wrot 5.21773E+01 5.03611E-05  <-- qu'avec diffusion du moment cinétique
.................................

 4.685000000000000E+03 10  1     31 T
 3.761741E+00-3.270666E-06 2.093004E-06 1.000000E+00 2.259322E-02 7.173108E-01
-1.000000E+02
```

```
 H1  3.22007E-01 7.30094E-01
He3  6.46882E-06 8.19412E-05
He4  6.57151E-01 2.51936E-01
C12  2.05912E-05 3.04480E-03
C13  5.63346E-06 3.64768E-05
N14  5.13761E-03 9.44078E-04
N15  2.25834E-07 3.69734E-06
O16  8.99719E-03 8.61328E-03
O17  7.99800E-04 3.47279E-06
Si28 5.87454E-03 5.24189E-03
Wrot 1.94190E+02 8.24763E-05  <-- qu'avec diffusion du moment cinétique
```

Les fichiers pour diagramme HR sont lus par la routine `lit_hr` du module `mod_exploit`.

C.3 Fichier ASCII des variables de la diffusion du moment cinétique

Le fichier ASCII des variables de la diffusion du moment cinétique est créé dans l'environnement du calcul suivant les circonstances décrites au §3.10 (Page 25). Il peut être exploité par le programme de dessin `des2k_rot` du sous-directory `EXPLOIT`, *cf.* §**??** (Page **??**).

`en-tête` :

- Lignes 1-4 : identification du modèle et physique utilisée.
- Ligne 5 : noms des variables dans l'ordre d'écriture.
- Ligne 6 : nombre, noms des éléments chimiques utilisés.
- Ligne 7 : nombre de couches, nombre total de variables, nombre de variables, d'éléments chimiques (redondant), identificateur de formalisme de diffusion de moment cinétique (3 pour Talon & Zhan 1997, 4 pour Mathis & Zahn 2004), indice de ^{4}He, numéro du modèle, indice de ^{7}Li.
- Ligne suivantes : `FORMAT(5es19.12)` variables et composition chimique.

Le modèle est listé du centre vers la surface. Exemple :

```
Fichier pour la diffusion du moment cinétique: 1.0_coeff_rota.dat
CESAM2k version V2.2.0 lagr colloc  2  3 pr no diffus, 18 Mai 2006 11h59
Physique utilisée: etat_eff, opa_yveline, conv_jmj, ppcno9, ctes_94, NACRE, diff_tz97
solaire_gn, lim_atm, hopf, pertm_ext, pertw_0, diffm_mp, difft_nu, diffw_mpz, diff_tz97
Variables : R, M, Omega, U, Theta, Psi, Lambda, Flux, Deff, Dh, Dv, T, ro, grad_mu, Xchim
 10  H1  He3  He4  C12  C13  N14  N15  O16  O17  Si28
       601        24        14        10         3         3        42         0
 0.000000000000E+00 0.000000000000E+00 2.492796313373E-06 1.222009447047E-15-2.215093172271E-11
 4.146564017724E-13 2.345111261140E-11-9.012551719834E-35 2.621399230830E+00 3.397109673134E+00
 2.621399230830E+00 1.340098717163E+07 8.330731227990E+01 1.426753865039E-02 6.871377248045E-01
 5.018205056432E-05 2.922381643859E-01 1.473316033117E-05 4.025153911526E-06 5.082420872692E-03
 2.299827930187E-07 9.638362729473E-03 6.971562756039E-06 5.827185297122E-03
 1.644625353770E-02 2.636507234666E-04 2.492796313373E-06 1.222009447047E-15-2.215093172271E-11
 4.146564017724E-13 2.345111261140E-11-9.012551719834E-35 2.621399230830E+00 3.397109673134E+00
 2.621399230830E+00 1.340098717163E+07 8.330731227990E+01 1.426753865039E-02 6.871377248045E-01
 5.018205056432E-05 2.922381643859E-01 1.473316033117E-05 4.025153911526E-06 5.082420872692E-03
 2.299827930187E-07 9.638362729473E-03 6.971562756039E-06 5.827185297122E-03
 2.464852077382E-02 8.840621368772E-04 2.492616591912E-06 3.215689566074E-15-4.144404321209E-11
 1.675631823126E-12 4.481166434756E-11-5.704344611195E-34 2.614052191228E+00 3.746479760886E+01
 2.614052191228E+00 1.334650928015E+07 8.276105949309E+01 8.706470878900E-03 6.872913782734E-01
 5.143754364837E-05 2.920832735684E-01 1.464390266467E-05 4.002793748349E-06 5.082554714526E-03
 2.301323508453E-07 9.638504967128E-03 6.795246250354E-06 5.827178857934E-03

.................................
.................................

 8.794865539306E-01 9.999999998874E-01 2.361680091400E-06-5.768386555867E-05 2.367155329527-282
-6.926405852191-289 3.012743146671-282-1.572130233500E-28 1.000000000000E+13 1.000000000000E+13
 1.000000000000E+13 9.146301617794E+03 4.727352051188E-07 0.000000000000E+00 6.999999992655E-01
 8.827001090114E-05 2.799117307236E-01 3.424868374763E-03 4.127760067609E-05 1.059003261745E-03
 4.167448298824E-06 9.640129747024E-03 3.902528976547E-06 5.826651038552E-03
 8.795031344698E-01 1.000000000000E+00 2.361680091400E-06-5.770215872484E-05 6.993868019057-283
-2.308801950730-289 1.022180710478-282-1.515562617419E-28 1.000000000000E+13 1.000000000000E+13
 1.000000000000E+13 8.883218491357E+03 4.555466484484E-07 0.000000000000E+00 6.999999992655E-01
 8.827001090114E-05 2.799117307236E-01 3.424868374763E-03 4.127760067609E-05 1.059003261745E-03
 4.167448298824E-06 9.640129747024E-03 3.902528976547E-06 5.826651038552E-03
```

Appendix D

Liste détaillée des modules

Afin de faciliter la lecture des algorithmes de Cesam2k20, on donne la liste détaillée des variables et routines des modules, en indiquant l'endroit de leur initialisation; celles-ci sont souvent effectuées dans des routines dépendantes, *cf.* §7.1 (Page 135), par raison de concision, seuls les noms des routines génériques sont mentionnés. Les routines étant par ailleurs largement décrites, elles ne sont ci-après que citées. Chaque routine `PRIVATE` et/ou `PUBLIC` est introduite sous la forme d'`INCLUDE` dans son module d'appartenance. Ci-après, la description des fonctions des variables logiques correspond à leur valeur .TRUE. L'ordre de présentation est celui de la compilation indiqué dans le fichier `list_cesam`,*cf.* §**??** (Page **??**), du sous-dirctory `SOURCE`.

D.1 Module mod_kind

Le module `mod_kind` regroupe les types des variables.

- Paramètres PUBLIC :

  dp : Définit les variables en double précision.
  sp : Définit les variables en simple précision.

D.2 Module mod_numerique

Le module `mod_numerique` regroupe les routines purement numériques.

- Variable PUBLIC logical :

  no_croiss : La suite des abscisses n'est pas strictement croissante, ou encore une autre difficulté est survenue lors de l'éxécution.

- Routines PRIVATE : **bval0, colpnt, difdiv, horner, schu58_n.**
- Routines PUBLIC : **arb_rom, boite, box, bsplddn, bspldn, bsp_dis, bvald, bvall, coll, delete_doubles, fermi_dirac, gauss_band, genere_bases, intgauss, linf, matinv, max_local, min_max, neville, newspl, newton, noedif, noein, noeud, noeu_dis, pause, polyder, shell, sum_n, zoning**

D.3 Module mod_donnees

Le module `mod_donnees` regroupe l'ensemble des quantités fixes au cours de l'évolution.

- Paramètres PUBLIC double précision :

d_conv : Coefficient de diffusion dans les zones mélangées.

dtmin : Pas temporel minimum en Myrs.

dx_tams : Précision sur l'abondance en X à la fin de la séquence principale d'âge zéro.

phi : $d\ln\rho/d\ln$ţ.

x_tams : Abondance en X définissant la fin de la séquence principale d'âge zéro.

- Paramètres PUBLIC integer :

n_min : Nombre minimum de couches.

pnzc : Nombre maximum de zones convectives pouvant être prises en compte.

r_qs : Ordre des équations différentielles des équations d'équilibre quasi-statique.

- Paramètres PUBLIC character :

version : Numéro de version lu par un INCLUDE du fichier `journal`, *cf.* §?? (Page ??).

- Variables PUBLIC double précision :

ab_ini : Abondances initiales, initialisé dans `nuc.f90`.

ab_min : Abondances minimales, initialisé dans `nuc.f90`.

nucleo : Masses atomiques des isotopes, initialisé dans `tabul_nuc.f90`.

rot_min : Valeurs négligeables des variables de la diffusion du moment cinétique, initialisé dans `lit_nl.f90`.

xvent : Composition chimique du vent, initialisé dans `vent.f90`.

zi : Charges des isotopes, initialisé dans `tabul_nuc.f90`.

abon_m : Abondances initiales par masse des éléments chimiques, initialisé dans `abon_ini.f90`.

aal27 : Masse atomique du ^{27}Al, initialisé dans `taux_nuc.f90`.

abe7 : Masse atomique du ^{7}Be, initialisé dans `taux_nuc.f90`.

abe9 : Masse atomique du ^{9}Be, initialisé dans `taux_nuc.f90`.

abl1 : Masse atomique du ^{11}B, initialisé dans `taux_nuc.f90`.

ac12 : Masse atomique du ^{12}C, initialisé dans `taux_nuc.f90`.

ac13 : Masse atomique du ^{13}C, initialisé dans `taux_nuc.f90`.

afe56 : Masse atomique du ^{56}Fe, initialisé dans `taux_nuc.f90`.

af18 : Masse atomique du ^{18}F, initialisé dans `taux_nuc.f90`.

af19 : Masse atomique du ^{19}F, initialisé dans `taux_nuc.f90`.

agemax : Age maximum à atteindre, lu dans `lit_nl.f90`.

ah : Masse atomique du ^{1}H, initialisé dans `taux_nuc.f90`.

ah2 : Masse atomique du ^{2}H, initialisé dans `taux_nuc.f90`.

ahe3 : Masse atomique du ^{3}He, initialisé dans `taux_nuc.f90`.

ahe4 : Masse atomique du ^{4}He, initialisé dans `taux_nuc.f90`.

ali6 : Masse atomique du ^{6}Li, initialisé dans `taux_nuc.f90`.

ali7 : Masse atomique du ^{7}Li, initialisé dans `taux_nuc.f90`.

alpha : Longueur de mélange, lu dans `lit_nl.f90`.

amg23 : Masse atomique du ^{23}Mg, initialisé dans `taux_nuc.f90`.

amg24 : Masse atomique du ^{24}Mg, initialisé dans `taux_nuc.f90`.

amg25 : Masse atomique du ^{25}Mg, initialisé dans `taux_nuc.f90`.

amg26 : Masse atomique du ^{26}Mg, initialisé dans `taux_nuc.f90`.

amu : Masse atomique unité m_u, initialisé dans `taux_nuc.f90`.

an : Masse atomique du neutron, initialisé dans `taux_nuc.f90`.

ana23 : Masse atomique du ^{23}Na, initialisé dans `taux_nuc.f90`.

ane20 : Masse atomique du ^{20}Ne, initialisé dans `taux_nuc.f90`.

ane21 : Masse atomique du ^{21}Ne, initialisé dans `taux_nuc.f90`.

ane22 : Masse atomique du ^{22}Ne, initialisé dans `taux_nuc.f90`.

an13 : Masse atomique du ^{13}N, initialisé dans `taux_nuc.f90`.

an14 : Masse atomique du ^{14}N, initialisé dans `taux_nuc.f90`.

an15 : Masse atomique du ^{15}N, initialisé dans `taux_nuc.f90`.

ap : Masse atomique du proton, initialisé dans `taux_nuc.f90`.

ap31 : Masse atomique du ¶31, initialisé dans `taux_nuc.f90`.

ao16 : Masse atomique du Ø16, initialisé dans `taux_nuc.f90`.

ao17 : Masse atomique du Ø17, initialisé dans `taux_nuc.f90`.

ao18 : Masse atomique du Ø18, initialisé dans `taux_nuc.f90`.

asi28 : Masse atomique du ^{28}Si, initialisé dans `taux_nuc.f90`.

as32 : Masse atomique du ^{32}S, initialisé dans `taux_nuc.f90`.

aradia : Constante de la radiation a, initialisé dans `ini_ctes.f90`.

clight : Célérité de la lumière c, initialisé dans `ini_ctes.f90`.

cpturb : Paramètre de pression turbulente, lu dans `lit_nl.f90`.

ctel : Facteur de répartition en luminosité, initialisé dans `cesam.f90`.

ctem : Facteur de répartition en masse, initialisé dans `cesam.f90`.

ctep : Facteur de répartition en pression, initialisé dans `cesam.f90`.

cter : Facteur de répartition en rayon, initialisé dans `cesam.f90`.

ctet : Facteur de répartition en température, initialisé dans `cesam.f90`.

dpsi : Taux de variation maximale pour la constante de répartition entaînant une modification du nombre de couches `n_qs` dans le cas standard d'une grille ajustable, initialisé dans `cesam.f90`.

dn_fixe : Taux de variation maximale pour la constante de répartition entaînant une modification du nombre de couches `nc_fixe` dans le cas d'une grille fixe, initialisé dans `cesam.f90`.

dtlist : Intervalle de temps entre deux listings complets, lu dans `lit_nl.f90`.

dtmax : Pas temporel maximal, initialisé dans `cesam.f90`.

dt0 : Pas temporel initial, initialisé dans `cesam.f90`.

d_grav : Variation maximale de l'énergie graviphique sur un pas temporel, initialisé dans `cesam.f90`.

d_turb : Coefficient de diffusion turbulente, lu dans `lit_nl.f90`.

echarg : Charge de l'électron, initialisé dans `ini_ctes.f90`.

eve : Energie de l'électronvolt, initialisé dans `ini_ctes.f90`.

fesh_sol : Valeur du [Fe/H] solaire, initialisé dans `ini_ctes.f90`.

g : Constante de la gravitation G, initialisé dans `ini_ctes.f90`.

gmsol : Produit $G \times M_\odot$, initialisé dans `ini_ctes.f90`.

granr : Constante des gaz parfaits $\mathcal{R}$, initialisé dans `ini_ctes.f90`.

he_core : Masse du noyau d'hélium à partir de laquelle l'évolution sera arrêtée, lu dans `lit_nl.f90`.

hhe_core : Masse réduite du noyau d'hélium à partir de laquelle l'évolution sera arrêtée , initialisé dans `lit_nl.f90`.

hpl : Constante de Plank h, initialisé dans `ini_ctes.f90`.

kbol : Constante de Boltzman k, initialisé dans `ini_ctes.f90`.

lbol0 : Zéro des magnitudes bolométriques, initialisé dans `ini_ctes.f90`.

li_ini : Abondance de ^{7}Li dans le noyau d'un modèle de séquence principale d'âge zéro homogène, initialisé dans `abon_ini.f90`.

lnt_stop : L'évolution est arrêtée lorsque cette température effective, en $\ln$, est franchie par valeurs croissantes ou décroissantes, initialisé dans `lit_nl.f90` .

ln_Tli : Température au dessus de laquelle il n'y a pas de lithium inital pour le modèle de séquence principale d'âge zéro, initialisé dans `abon_ini.f90`.

ln10 : $\ln(10)$, initialisé dans `ini_ctes.f90`.

loc_zc : Paramètre de localisarion des limites zones radiatives / zones convectives, initialisé dans `cesam.f90`.

log_teff : L'évolution est arrêtée lorsque cette température effective est franchie par valeurs croissantes ou décroissantes, lu dans `lit_nl.f90`.

lsol : Luminosité solaire $L_\odot$, initialisé dans `ini_ctes.f90`.

mdot : Taux de perte de masse, lu dans `lit_nl.f90`.

me : Masse de l'électron m_e, initialisé dans `ini_ctes.f90`.

msol : Masse solaire $M_\odot$, initialisé dans `ini_ctes.f90`.

mterre : Masse de la terre, initialisé dans `ini_ctes.f90`.

mtot : Masse initiale du modèle, lu dans `lit_nl.f90`.

ovshti : Taux d'overshoot inférieur, lu dans `lit_nl.f90`.

ovshts : Taux d'overshoot supérieur, lu dans `lit_nl.f90`.

pi : π, initialisé dans `ini_ctes.f90`.

precit : Paramètre de précision de l'intégration temporelle de la composition chimique, initialisé dans `cesam.f90`.

precix : Paramètre de précision de l'intégration spatiale de l'équilibre quasi-statique, initialisé dans `cesam.f90`.

p_pertw : Paramètre de perte de moment cinétique, lu dans `lit_nl.f90`.

re_nu : Taux de turbulence radiative, lu dans `lit_nl.f90`.

ro_test : Valeur minimale de la densité en dessous de laquelle la variation d'énergie graviphique n'est plus considérée comme significative, initialisé dans `cesam.f90`.

rsol : Rayon solaire $R_\odot$, initialisé dans `ini_ctes.f90`.

secon6 : Nombre de secondes par million d'années, initialisé dans `ini_ctes.f90`.

sigma : Constante de Stéfan σ, initialisé dans `ini_ctes.f90`.

tau_max : Profondeur optique de raccord atmosphère/enveloppe, lu dans `lit_nl.f90`.

t_inf : Température minimale de tabulation des réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

t_sup : Température maximale de tabulation des réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

t_stop : Température centrale arrêtant le calcul de l'évolution, lu dans `lit_nl.f90`.

ua : Unité astronomique, initialisé dans `ini_ctes.f90`.

w_form : Facteur entre 0 et 1 permettant de modifier le profil de la vitesse angulaire initiale, initialisé dans `initialise_u.f90`.

w_rot : Vitesse angulaire initiale, lu dans `lit_nl.f90`.

x0 : Abondance initiale par masse d'hydrogène X, lu ou déterminé dans `lit_nl.f90`.

x_stop : Abondance centrale d'hydrogène arrêtant le calcul de l'évolution, lu dans `lit_nl.f90`.

y0 : Abondance initiale par masse d'hélium Y, lu dans `lit_nl.f90`.

zsx_sol : Z/X solaire, initialisé dans `ini_ctes.f90`.

zsx0 : Rapport initial Z/X, lu ou initialisé dans `lit_nl.f90` .

z0 : Abondance initiale par masse des métaux Z, initialisé dans `lit_nl.f90`.

- Variables PUBLIC simple précision :

dh : Espace en hauteur entre cadres des dessins, initialisé dans `des.f90`.

dl : Espace en largeur entre cadres des dessins, initialisé dans `des.f90`.

h : Hauteur des cadres des dessins, initialisé dans `des.f90`.

ld : Largeur des cadres des dessins, initialisé dans `des.f90`.

fesh_des : Cible en [Fe/H] en surface, initialisé dans `des.f90`.

l_des : Cible en $L/L_\odot$, initialisé dans `des.f90`.

teff_des : Cible en température, initialisé dans `des.f90`.

logteff_max : Limite à gauche des abscisses du cadre du diagramme HR, initialisé dans `des.f90`.

logteff_min : Limite à droite des abscisses du cadre du diagramme HR, initialisé dans `des.f90`.

logl_max : Limite supérieure des ordonnées du cadre du diagramme HR, initialisé dans `des.f90`.

logl_min : Limite inférieure des ordonnées du cadre du diagramme HR, initialisé dans `des.f90`.

xleft : Limite à droite des dessins, initialisé dans `des.f90`.

ybot : Limite inférieure des dessins, initialisé dans `des.f90`.

y_age : Ordonnées pour l'écriture de l'âge au dessus des cadres des dessins, initialisé dans `des.f90`.

- Variables PUBLIC integer :

Kdes_rot : Flag d'identification du type de dessin pour la rotation, initialisé dans `lit_nl`.

Krot : Flag d'identification du type de rotation, initialisé dans `lit_nl`.

ife56 : Indice de l'isotope ^{56}Fe, initialisé dans `nuc.f90`.

ihe4 : Indice de l'isotope ^{4}He, initialisé dans `nuc.f90`.

iLi7 : Indice de l'isotope ^{7}Li, initialisé dans `nuc.f90`.

ini0 : Nombre maximum d'itérations globales avec réestimation de la composition chimique et de la délimitation des limites zones radiatives / zones convectives, initialisé dans `cesam.f90`.

lpg : Indice de la variable pression gazeuse, initialisé dans `cesam.f90`.

i_ex : Indice de l'élément fictif, initialisé dans `nuc.f90`.

m_ch : Ordre des B-splines d'interpolation de la composition chimique, initialisé dans `cesam.f90`.

m_ptm : Ordre des B-splines d'interpolation de la perte de masse, initialisé dans `cesam.f90`.

m_qs : Ordre des B-splines pour la résolution de l'équilibre quasi-statique, initialisé dans `cesam.f90`.

m_rot : Ordre des B-splines d'interpolation du moment cinétique, initialisé dans `cesam.f90`.

m_tds : Ordre des B-splines d'interpolation de l'énergie graviphique, initialisé dans `cesam.f90`.

nb_max_modeles : Nombre maximum de modèles à calculer, lu dans `lit_nl.f90`.

nchim : Nombre d'isotopes, initialisé dans `nuc.f90`.a

ne : Nombre de variables pour l'équilibre quasi-statique,initialisé dans `cesam.f90`.

nrl : Nombre de coefficients pour la rotation, initialisé dans `lit_nl.f90`.

nrot : Nombre de variables pour la diffusion du moment cinétique, initialisé dans `lit_nl.f90`.

n_atm : Nombre de couches dans l'atmosphère, initialisé dans `cesam.f90`.

n_max : Nombre maximum de couches pour l'équilibre quasi-statique, initialisé dans `cesam.f90`.

ordre : Ordre d'intégration des équations d'évolution de la composition chimique sans diffusion, initialisé dans `cesam.f90`.

ord_qs : Ordre des B-splines pour l'intégration des équations d'équilibre quasi-statique, initialisé dans `cesam.f90`.

- Variables PUBLIC logical :

ajuste : On ajustera à 1% près lesvaleurs de sortie, par exemple l'abondance d'hydrogène au centre, initialisé dans `cesam.f90` suivant le type de précision requis.

all_output : On garde dans l'environnement tous les fichiers binaires de reprise dont le nom comporte le numéro du modèle; initialisé dans `lit_nl.f90`.

all_rep : On garde dans l'environnement tous les fichiers ASCII dont le nom comporte le numéro du modèle; initialisé dans `lit_nl.f90`.

diffusion : On tient compte de la diffusion microscopique des isotopes, lu dans `lit_nl.f90`.

en_masse : Utilisation des variables lagrangiennes, lu dans `lit_nl.f90`.

garde_xish : On conserve les rapports métal/hydrogène de la mixture initiale, lu dans `lit_nl.f90`.

grad_ovi : On utilise le gradient adiabatiaque dans les extensions inférieures des zone convective; initialisé dans `lit_nl.f90`.

grad_ovs : On utilise le gradient adiabatiaque dans les extensions supérieures des zone convective; initialisé dans `lit_nl.f90`.

grille_fixe : La grille d'interpolation de la composition chimique est fixe, lu dans `lit_nl.f90`.

He_ajuste : On ajustera la limite du noyau d'hélium avant de sortir, initialisé dans `lit_nl.f90`.

jpz : Utilisation du formalisme de pénétration convective de JP. Zahn, lu dans `lit_nl.f90`.

kipp : Utilisation de l'approximation de Kippenhahan, lu dans `lit_nl.f90`.

ledoux : Utilisation du critère de convection de Ledoux, lu dans `lit_nl.f90`.

lim_ro : Utilisation de la limite en densité pour la restitution de l'atmosphère, lu dans `lit_nl.f90`.

lim_jpz : Utilisation de la limite de JpZh pour la diffusion du moment angulaire, lu dans `lit_nl.f90`.

lisse : Il y aura lissage par `contour` des profils des abondances, initialisé dans `cesam.f90` suivant le type de précision requis.

mitler : Utilisation du formalisme de Mitler pour le calcul de l'effet d'écran, lu dans `lit_nl.f90`.

modif_chim : On personnalisera la composition chimique initiale, lu dans `lit_nl.f90`.

mu_saha : Pour le calcul des coefficients de la diffusion du moment cinétique, le poids moléculaire moyen sera calculé en tenant compte des divers taux d'ionisation calculés par `saha.f90`, initialisé dans `cesam.f90` suivant le type de précision requis.

mvt_dis : On tiendra compte des déplacements des discontinuités pour la détermination de la composition chimique, initialisé dans `cesam.f90`.

pturb : On tiendra compte de la pression turbulente, initialisé dans `lit_nl`.

rep_atm : On utilisera un fichier binaire d'atmosphère, s'il existe, initialisé dans `cesam.f90`.

rot_solid : On supposera la rotation solide, lu dans `lit_nl.f90`.

t_ajuste : On ajustera la température centrale avant de sortir,initialisé dans `lit_nl.f90`.

x_ajuste : On ajustera l'abondance centrale d'hydrogène avant de sortir, initialisé dans `lit_nl.f90`.

- Variables PUBLIC character :

precision : Désignation du type de calcul, lu dans `lit_nl.f90`.

nom_elem : Noms des isotopes utilisés, initialisé dans `tabul_nuc.f90`.

nom_rot : Noms des variables utilisées pour la diffusion du moment cinétique, initialisé dans `lit_nl.f90`.

arret : Désignation du type d'arrêt, lu dans `lit_nl.f90`.

nom_xheavy : Nom de l'élément fictif, initialisé dans `nuc.f90`.

unit : Nom de l'unité utilisée pour la donnée de la vitesse angulaire initiale, initialisé dans `lit_nl.f90`.

langue : Nom de la langue utilisée pour les commentaires, initialisé dans `cesam.f90` .

nom_atm : Nom du type d'atmosphère, lu dans `lit_nl.f90`.

nom_abon : Nom des abondances initiales, lu dans `lit_nl.f90`.

nom_conv : Nom du type de convection, lu dans `lit_nl.f90`.

nom_ctes : Nom du groupe des constantes physiques, lu dans `lit_nl.f90`.

nom_des : Nom du type de dessin, lu dans `lit_nl.f90`.

nom_diffm : Nom du type de coefficients de diffusion microscopique, lu dans `lit_nl.f90`.

nom_diffw : Nom du type de coefficients de diffusion de moment cinétique, lu dans `lit_nl.f90`.

nom_difft : Nom du type de coefficients de diffusion turbulente à utiliser, lu dans `lit_nl.f90`.

nom_etat : Nom du type d'Equation of State, lu dans `lit_nl.f90`.

nom_frad : Nom du type d'accélérations radiatives, lu dans `lit_nl.f90`.

nom_nuc : Nom du type de réseau de réactions thermonucléaires, lu dans `lit_nl.f90`.

nom_nuc_cpl : Nom du type de compilation de réactions thermonucléaires, lu dans `lit_nl.f90`.

nom_output : Nom du type de sortie ASCII, lu dans `lit_nl.f90`.

nom_pertm : Nom du type de perte de masse, lu dans `lit_nl.f90`.

nom_pertw : Nom du type de perte de moment cinétique, lu dans `lit_nl.f90`.

nom_tdetau : Nom du type de loi $T(\tau)$, lu dans `lit_nl.f90`.

nom_thw : Nom de la théorie de la diffusion du moment du moment cinétique, lu dans `lit_nl.f90`.

nom_fich2 : Nom du fichier de sortie `mon_modele.lis`, lu dans `lit_nl.f90`.

thw : Noms des théories de la diffusion du moment du moment cinétique implantées dans Cesam2k20, initialisé dans `lit_nl.f90`.

f_eos : Nom du fichier d'Equation of State, lu dans `lit_nl.f90`.

f_opa : Nom du fichier d'opacité, lu dans `lit_nl.f90`.

nom_opa : Nom du type d'opacité, lu dans `lit_nl.f90`.

source : Nom de la source des constantes physiques, initialisé dans `ini_ctes.f90`.

methode : Description de la méthode de calcul, initialisé dans `cesam.f90`.

device : Nom du device de dessin, initialisé dans `des.f90`.

nom_chemin : Nom du chemin du directory des données physiques, lu dans `lit_nl.f90`.

- Routines PUBLIC : **lit_nl, ini_ctes, print_ctes.**
- Routines PRIVATE : **ctes_85, ctes_94, ctes_94m.**

D.4 Module mod_variables

Ce module regroupe les routines d'usage général et les quantités variant au cours d'une évolution. Par exemple, le vecteur des variables quasi-statiques, ou encore le vecteur de composition chimique.

- Variables PUBLIC double précision :

bp : Table des coefficients des B-splines des variables quasi-statiques, calculée dans `coll_qs.f90`.

bp_t : Table des coefficients des B-splines des variables quasi-statiques au pas temporel précédent, réinitialisé dans `update.f90`.

chim : Table des coefficients des B-splines de la composition chimique, calculé dans `evol.f90` ou `resout_diff_chim.f90`.

chim_t : Table des coefficients des B-splines de la composition chimique au pas temporel précédent, réinitialisé dans `update.f90`.

old_ptm : Tables des coefficients des B-splines des masses, calculé dans `pertm.f90`.

rota : Table des coefficients des B-splines de la rotation, calculé dans `resout_rota.f90`.

rota_t : Table des coefficients des B-splines de la rotation au pas temporel précédent, réinitialisé dans `update.f90`.

tds : Table des coefficients des B-splines de l'énergie graviphique, calculé dans `cesam.f90`.

tds_t : Table des coefficients des B-splines de l'énergie graviphique au pas temporel précédent, réinitialisé dans `update.f90`.

mc : Table des masses pour l'interpolation de la composition chimique, calculé dans `evol.f90` ou `diffus.f90`.

mct : Vecteur nodal des masses pour l'interpolation de la composition chimique, calculé dans `evol.f90` ou `diffus.f90`.

mc_fixe : Table des masses de la grille fixe, calculé dans `evol.f90`.

mc_t : Table des masses pour l'interpolation de la composition chimique au pas temporel précédent, réinitialisé dans `update.f90`.

mct_t : Vecteur nodal des masses pour l'interpolation de la composition chimique au pas temporel précédent, réinitialisé dans `update.f90`.

mrot : Table des masses pour l'interpolation du moment cinétique, calculé dans `diffus.f90`.

mrott : Vecteur nodal des masses pour l'interpolation du moment cinétique, calculé dans `diffus.f90`.

mrott_t : Vecteur nodal des masses pour l'interpolation du moment cinétique au pas temporel précédent, réinitialisé dans `update.f90`.

mrot_t : Table des masses pour l'interpolation du moment cinétique au pas temporel précédent, réinitialisé dans `update.f90`.

m23 : Table de masses simplifiant l'interpolation inverse, calculé dans `resout.f90`.

m23_t : Table de masses simplifiant l'interpolation inverse au pas temporel précédent, réinitialisé dans `update.f90`.

q : Table des abscisses entières pour linterpolation des variables quasi-statiques, calculé dans `lim_zc.f90`.

qt : Vecteur nodal des abscisses entières, calculé dans `lim_zc.f90`.

q_t : Table des abscisses entières au pas temporel précédent, réinitialisé dans `update.f90`.

qt_t : Vecteur nodal des abscisses entières au pas temporel précédent, réinitialisé dans `update.f90`.

r2 : Table de rayons simplifiant l'interpolation inverse, calculé dans `.f90`.

r2_t : Table de rayons simplifiant l'interpolation inverse au pas temporel précédent, réinitialisé dans `update.f90`.

xl : Abscisses des limites pour l'équilibre quasi-statique, calculé dans `coll_qs.f90`.

xt_ptm : Vecteur nodal pour la tabulation de la perte de masse, calculé dans `pertm.f90`.

xt_tds : Vecteur nodal pour la tabulation de l'énergie graviphique, calculé dans `cesam.f90`.

xt_tds_t : Vecteur nodal pour la tabulation de l'énergie graviphique au pas temporel précédent, réinitialisé dans `update.f90`.

x_ptm : Table des masses pour la tabulation de la perte de masse, calculé dans `.f90`.

x_tds : Table des masses pour la tabulation de l'énergie graviphique, calculé dans `.f90`.

x_tds_t : Table des masses pour la tabulation de l'énergie graviphique au pas temporel précédent, réinitialisé dans `update.f90`.

x_planet : Composition chimique des planétoîdes, initialisé dans `planetoides.f90`.

r_zc_conv : Rayons des limites des zones convectives, calculé dans `lim_zc.f90`.

m_zc : Masses aux limites des zones convectives, calculé dans `lim_zc.f90`.

m_zc_t : Masses aux limites des zones convectives au pas temporel précédent, réinitialisé dans `update.f90`.

r_zc : Rayons des zones convectives, calculé dans `lim_zc.f90`.

r_zc_t : Rayons des zones convectives au pas temporel précédent, réinitialisé dans `update.f90`.

r_ov : Rayons des zones convectives avec overshoot, calculé dans `lim_zc.f90`.

r_ov_t : Rayons des zones convectives avec overshoot au pas temporel précédent, réinitialisé dans `update.f90`.

age : Age du modèle, calculé dans `cesam.f90`.

c_iben : Constante de Iben , calculé dans `cesam.f90`.

mstar : Masse du modèle, calculé dans `atm.f90`.

mstar_t : Masse du modèle au pas temporel précédent, réinitialisé dans `update.f90`.

mw_tot : Moment cinétique total, calculé dans `resout.f90`.

mw_tot_t : Moment cinétique total au pas temporel précédent, réinitialisé dans `update.f90`.

psi0 : Constante de répartition, calculé dans `resout.f90`

rstar : Rayon de l'étoile, calculé dans `atm.f90`.

wrot : Vitesse angulaire, calculé dans `resout.f90`.

wrot_t : Vitesse angulaire au pas temporel précédent, réinitialisé dans `update.f90`.

- Variables PUBLIC integer :

jlim : Indices des limites zones radiatives / zones convectives, calculé dans `lim_zc.f90`.

jlim_t : Indices des limites zones radiatives / zones convectives au pas temporel précédent, réinitialisé dans `update.f90`.

dim_ch : Dimension de l'espace des splines pour la composition chimique, réinitialisé dans `evol.f90`.

dim_qs : Dimension de l'espace des splines pour les variables de structure, réinitialisé dans `lim_zc.f90`.

dim_rot : Dimension de l'espace des splines pour la diffusion du moment cinétique, réinitialisé dans `evol.f90`.

id_conv : Indices des débuts des zones convectives, réinitialisé dans `evol.f90`.

if_conv : Indices des fins des zones convectives, réinitialisé dans `evol.f90`.

knot : Nombre de points du vecteur nodal pour l'interpolation des variables quasi-statiques, calculé dans `coll.f90`.

knotc : Nombre de points du vecteur nodal pour l'interpolation de la composition chimique, calculé dans `evol.f90` ou `resout_diff_chim`.

knotc_t : Nombre de points du vecteur nodal pour l'interpolation de la composition chimique au pas temporel précédent, réinitialisé dans `update.f90`.

knot_ptm : Nombre de points du vecteur nodal pour l'interpolation de la perte de masse, calculé dans `pertm.f90`.

knotr : Nombre de points du vecteur nodal pour l'interpolation du vecteur de mélange, calculé dans `diffus.f90`.

knotr_t : Nombre de points du vecteur nodal pour l'interpolation du vecteur de mélange, au pas temporel précédent, réinitialisé dans `update.f90`.

knot_t : Nombre de points du vecteur nodal pour l'interpolation des variables quasi-statiques au pas temporel précédent, réinitialisé dans `update.f90`.

knot_tds : Nombre de points du vecteur nodal pour l'interpolation de l'énergie graviphique, calculé dans `cesam.f90`.

knot_tds_t : Nombre de points du vecteur nodal pour l'interpolation de l'énergie graviphique au pas temporel précédent, réinitialisé dans `update.f90`.

lim : Indices des limites zones radiatives / zones convectives, calculé dans `lim_zc.f90`.

lim_t : Indices des limites zones radiatives / zones convectives au pas temporel précédent, réinitialisé dans `update.f90`.

model_num : Indice du dernier modèle calculé depuis le début de l'évolution compte tenu des reprises, réinitialisé dans `cesam`.

nb_modeles : Nombre de modèles calculés au cours du présent run, réinitialisé dans `cesam`.

nc_fixe : Nombre de points de la grille fixe d'interpolation de la composition chimique, calculé dans `evol.f90`.

n_ch : Nombre de points de la grille d'interpolation de la composition chimique, calculé dans `evol.f90`.

n_ch_t : Nombre de points de la grille d'interpolation de la composition chimique au pas temporel précédent, réinitialisé dans `update.f90`.

n_ptm : Nombre de points de la grille d'interpolation de la perte de masse, calculé dans `pertm.f90`.

n_qs : Nombre de points de la grille d'interpolation de l'équilibre quasi-statique, calculé dans `lim_zc.f90`.

n_qs_t : Nombre de points de la grille d'interpolation de l'équilibre quasi-statique au pas temporel précédent, réinitialisé dans `update.f90`.

n_rot : Nombre de points de la grille d'interpolation de la rotation, calculé dans `diffus.f90`.

n_rot_t : Nombre de points de la grille d'interpolation de la rotation au pas temporel précédent, réinitialisé dans `update.f90`.

n_tds : Nombre de points de la grille d'interpolation de l'énergie graviphique, calculé dans `.f90`.

n_tds_t : Nombre de points de la grille d'interpolation de l'énergie graviphique au pas temporel précédent, réinitialisé dans `update.f90`.

- Variables PUBLIC logical :

lconv : Début d'une zone convective, calculé dans `lim_zc.f90`.

lconv_t : Début d'une zone convective au pas temporel précédent , réinitialisé dans `update.f90`.

lhe_stop : Arrêt lorsque le cœur d'hélium a atteint la masse requise, calculé dans `cesam.f90`.

lt_stop : Arrêt lorsque la température centrale a atteint la valeur requise, calculé dans `.f90`.

lx_stop : Arrêt lorsque l'abondance centrale d'hydrogène a atteint la valeur requise, calculé dans `cesam.f90`.

tot_conv : Le modèle est totalement convectif, calculé dans `lim_zc.f90`.

tot_rad : Le modèle est totalement radiatif, calculé dans `.f90`.

- Routines PUBLIC : **chim_gram, inter, sortie.**

D.5 Module mod_etat

Le module `mod_etat` regroupe les routines concernées par l'Equation of State.

- Routines PRIVATE : **etat_ceff, etat_eff, etat_gongl, etat_gong2, etat_mhd, etat_opal, etat_opalX, etat_opalZ.**
- Routines PUBLIC : **df_rotx, etat, saha.**

D.6 Module mod_opa

Le module mod_opa regroupe les routines concernant le calcul de l'opacité. La routine d'opacité z14xcotrin21.f90 n'est qu'associée au module en raison de sa programmation ancienne.

- Routines PRIVATE : **kappa_cond, opa_compton, opa_gong, opa_houdek9, opa_int_zsx, opa_opalCO, opa_opal2, opa_yveline, opa_yveline_lisse.**
- Routine PUBLIC : **opa.**

D.7 Module mod_conv

Le module mod_conv regroupe les routines concernant la convection.

- Routines PRIVATE : **conv_a0, conv_cgm_reza, conv_cm, conv_cml, conv_cm_reza, conv_jmj.**
- Routine PUBLIC : **conv.**

D.8 Module mod_atm

Le module mod_atm regroupe les routines de restitution de l'atmosphère.

- Variables PUBLIC double précision :

  bp_atm : Tableau des coefficients des B-splines d'interpolation de l'atmosphère, calculé dans coll_atm.f90.

  bp_atm_t : Tableau des coefficients des B-splines d'interpolation de l'atmosphère au pas temporel précédent, réinitialisé dans update.f90.

  dlpp_atm : Tableau des quotients $d\ln P_{\rm gaz}/d\ln P_{\rm tot}$, calculé dans atm.f90.

  m_atm : Tableau des masses, abscisses lagrangiennes ,calculé dans coll_atm.f90.

  p_atm : Tableau des pressions gazeuses, calculé dans atm.f90.

  pt_atm : Tableau des pressions totales, calculé dans atm.f90.

  r_atm : Tableau des rayons , calculé dans atm.f90.

  tau : Tableau des profondeurs optiques, calculé dans atm.f90.

  t_atm : Tableau des températures, calculé dans atm.f90.

  x_atm : Tableau des abscisses pour la restitution de l'atmosphère, calculé dans atm.f90.

  x_atm_t : Tableau des abscisses pour la restitution de l'atmosphère au pas temporel précédent, réinitialisé dans update.f90.

  xt_atm : Vecteur nodal des abscisses pour la restitution de l'atmosphère, calculé dans lim_atm.f90.

  tau_min : Epaisseur optique de la couche externe de l'atmosphère, initialisé dans tdetau.f90.

- Variables PRIVATE double précision :

  delfim : Pente de la fonction de répartition à gauche de la couche d'indice n23_atm, calculé dans lim_atm.f90.

  delfip : Pente de la fonction de répartition à droite de la couche d'indice n23_atm, calculé dans lim_atm.f90.

  ltaue : $\ln(\tau_{\rm min})$, initialisé dans lim_atm.f90.

ltauf : $\ln(\tau_{\max})$, initialisé dans `lim_atm.f90`.

tau_min : Epaisseur optique de la couche externe de l'atmosphère, initialisé dans `tdetau.f90`.

- Variables PRIVATE integer :

lpgt : Indice de la variable pression gazeuse, initialisé dans `lim_atm.f90`.

ne_atm : Nombre d'équations pour la restitution de l'atmosphère, initialisé dans `lim_atm.f90`.

n23_atm : Indice de la couche définissant le rayon bolométrique où $T(\tau*) = T_{\rm eff}$, initialisé dans `lim_atm.f90`.

- Variables PRIVATE logical :

rad : La loi T(tau) est purement radiative, initialisé dans `tdetau.f90`.

- Variable PRIVATE character :

nom_atm : Nom de la routine de loi $T(\tau)$ à utiliser, lu dans `lit_nl.f90`.

- Routines PRIVATE : **coll_atm, edding, eq_atm, hopf, k5750, k5777, lim_atm, lim_gongl, lim_taul, roger00, roger02, roger05, roger10a, taueff, trho**
- Routines PUBLIC : **atm, tdetau, thermo_atm.**

D.9 Module mod_nuc

Le module `mod_nuc` regroupe les routines concernant les réactions thermonucléaires.

- Paramètres PUBLIC integer :

m_temp : Ordre des splines pour l'interpolation des taux des réactions thermonucléaires.

niso_tot : Nombre maximum d'isotopes.

nreac_tot : Nombre maximum de réactions thermonucléaires.

- Paramètres PRIVATE integer :

nelem_ini : Nombre maximum d'éléments.

- Variables PUBLIC double précision :

taux_reac : Taux des réactions thermonucléaires, initialisé dans taux_nuc.f90.

ar : Masses réduites des cibles des réactions thermonucléaires, initialisé dans `taux_nuc.f90`.

q0 : Energie des réactions thermonucléaires, initialisé dans taux_nuc.f90.

temp, ttemp : Table des températures et vecteur nodal assicié pour l'interpolation des réactions thermonucléaires, initialisé dans `taux_nuc.f90`.

- Variables PRIVATE double précision :

ab : Tableau des abondances, initialisé dans `abon_ini.f90`.

abon_rela : Tableau des abondances relatives métal/Z, initialisé dans `abon_ini.f90`.

m : Masses atomiques, initialisé dans `abon_ini.f90`.

c : Charges des éléments chimiques, initialisé dans `abon_ini.f90`.

be7sbe9 : Rapport isotopique ^{7}Be/^{9}Be, initialisé dans `abon_ini.f90`.

be7sz : Rapport isotopique ^{7}Be/Z, initialisé dans `abon_ini.f90`.

c13sc12 : Rapport isotopique $^{13}C/^{12}C$, initialisé dans `abon_ini.f90`.

h2sh1 : Rapport isotopique $^{2}H/^{1}H$, initialisé dans `abon_ini.f90`.

he3she4 : Rapport isotopique $^{3}He/^{4}He$, initialisé dans `abon_ini.f90` .

he3she4z :Rapport isotopique $^{3}He/^{4}He$, avec ^{2}H primordial transformé en ^{3}He, initialisé dans `abon_ini.f90`.

li6sli7 : Rapport isotopique $^{6}Li/^{7}Li$, initialisé dans `abon_ini.f90`.

mg25smg24 : Rapport isotopique $^{25}Mg/^{24}Mg$, initialisé dans `abon_ini.f90`.

mg26smg25 :Rapport isotopique $^{26}Mg/^{25}Mg$, initialisé dans `abon_ini.f90`.

ne22sne20 : Rapport isotopique $^{22}Ne/^{20}Ne$, initialisé dans `abon_ini.f90`.

nl5snl4 : Rapport isotopique $^{15}N/^{14}N$, initialisé dans `abon_ini.f90`.

o17so16 : Rapport isotopique Ø17/Ø16, initialisé dans `abon_ini.f90`.

o18so16 : Rapport isotopique Ø18/Ø16, initialisé dans `abon_ini.f90`.

age_deb : Instant du début des chutes de planètoïdes, lu dans `planetoides.f90`.

age_fin : Instant de la fin des chutes de planètoïdes, lu dans `planetoides.f90`.

dt_planet : Pas temporel pour décrire les chutes de planètoïdes, lu dans `planetoides.f90`.

t_sup : Température maximale de tabulation des réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

mzc_ext : Masse de la zone convectiveexterne, réinitialisé dans `evol.f90`.

nuzc_ext : $M^{\frac{2}{3}}$ de la zone convective externe, réinitialisé dans `evol.f90`.

- Variables PRIVATE integer :

  izz : Table des charges des noyaux interagissant dans les réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

  i3al : Indice de la réaction 3α, initialisé dans `tabul_nuc.f90`.

- Variables PUBLIC integer :

  knot_temp : Dimension du vecteur nodal pour l'interpolation des réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

  nreac : Nombre de réactions thermonucléaires du réseau utilisé, initialisé dans `tabul_nuc.f90`.

  n_temp : Nombre de températures pour pour l'interpolation des réactions thermonucléaires, initialisé dans `tabul_nuc.f90`.

- Variable PRIVATE character :

  elem : Noms des éléments chimiques, initialisé dans `abon_ini.f90`.

- Routines PRIVATE : **iben, pp1, pp3, ppcno10BeBFe, ppcno10Fe, pcno10K, ppcno10, ppcno11, ppcno12Be, ppcno12BeBFe, ppcno12Li, ppcno12, ppcno3al2Ne, ppcno3a9, ppcno3aco, ppcno9, ppcno9Fe, rq_reac, tabul_nuc**

- Routines PUBLIC : **abon_ini, nuc, planetoides, taux_nuc, vent.**

D.10 Module mod_bp_for_alecian

Le module `mod_bp_for_alecian` regroupe les routines concernant le calcul des accélérations radiatives suivant le second formalisme de G.Alécian. Développé par B.Pichon, il constitue essentiellement une interface entre les routines fournies par G.Alécian et Cesam2k20. D'origine externe, les fonctions des composantes de ce module ne sont pas détaillées.

D.11 Module mod_evol

Le module `mod_evol` regroupe les routines concernant l'évolution temporelle avec et sans diffusion de la composition chimique et du moment cinétique.

- Paramètres PRIVATE double précision :

  eps : Petit écart permettant l'estimation numérique de dérivées.

  nu_min : Valeur minimale pour éviter $\nu = 0$.

  t_gab : Valeur minimale en dessous de laquelle il pourra y avoir mélange dans la routine `difft_gab.f90`.

  un_eps : Ecart pour l'estimation numérique de dérivées.

- Variables PRIVATE double précision :

  dr_zc : Variations des rayons des limites zones radiatives / zones convectivessur un pas temporel, réintialisé dans `evol.f90`.

  x_mix : Abscisses des limites zones radiatives / zones convectives, réintialisé dans `evol.f90`.

- Variables PRIVATE integer :

  idis : Table des indices des discontinuités, réinitialisé dans `evol.f90`.

  convd :Table des indices du début des discontinuités, réinitialisé dans `evol.f90`.

  convf : Table des indices de fin des discontinuités, réinitialisé dans `evol.f90`.

  ndis : Nombre de discontinuités, réinitialisé dans `evol.f90`.

  nzc : Nombre de zones convectives, réinitialisé dans `evol.f90`.

  n_mix : Nombre de points d'interpolation du vecteur de mélange, réinitialisé dans `evol.f90`.

- Variables PUBLIC logical :

  mix : Table logique des états de convection, réinitialisé dans `evol.f90`.

  lw_perte : Il y a perte de moment angulaire, initialisé dans `pertw.f90`.

- Routines PRIVATE : **alecianl, coeff_rota3, coeff_rota4, collision, coulomb, diffm, diffm_br, diffm_mp, difft, difft_gab, difft_nu, difft_sun, diffus, diffw, diffw_mpz, diffw_p03, eq_diff_chim, eq_diff_poisson, eq_diff_rota3, eq_diff_rota4, f_rad, initialise_poisson, initialise_u, initialise_w, lmix, pertw, pertw_loc, pertw_ptm, pertw_sch, resout_chim, resout_rota, resout_rota3, resout_rota4.**

- Routines PUBLIC : **coeff_rota, ecrit_rota, evol, initialise_rota, rk_imps.**

D.12 Module mod_static

Le module `mod_static` regroupe les routines concernées par la résolution des équations de l'équilibre quasi-statique.

- Paramètre PRIVATE character :

  nom_qs : Noms des variables quasi-statiques.

- Variables PRIVATE double précision :

  fac : Facteurs de répartition, calculé dans `lim_zc.f90`.

xcoll : Abscisses des points de collocation, réinitialisé dans `coll_qs.f90`.

- Routines PRIVATE : **coll_qs, dgrad, lim_zc, pertm_ext, pertm_msol, pertm_tot, pertm_waldron, static, static_m, static_r, update.**
- Routines PUBLIC : **pertm, resout, thermo.**

D.13 Module mod_cesam

Le module `mod_cesam` regroupe les routines concernant la gestion générale du calcul.

- Routines PRIVATE : **add_ascii, ascii, des, des_m, des_r, dnunl, list, output, osc_adia, osc_invers, osc_nadia**
- Routine PUBLIC : **cesam.**

D.14 Module mod_exploit

Le module `mod_exploit` regroupe des routines concernées par l'exploitation des résultats.

- Routine PRIVATE : **ascii, diffw_mpz, diffw_p03, osc_adia, osc_invers, osc_nadia.**
- Routines PUBLIC : **add_ascii, ctes_85, ctes_94, ctes_94m, diffw, ini_ctes, inter_atm, lit_binaire, lit_hr, lit_nl, min_max, min_max_cond, output, read_ascii, write_nl.**

Appendix E

Contenu des sous-directory

On donne la liste détaillée du contenu des sous-directory.

E.1 Sous-directory SOURCE

abon_ini.f : Initialisation des abondances, routine PRIVATE du module `mod_nuc`.

add_ascii.f : Complète l'ensemble des variables pour sorties ASCII, routine PRIVATE du module `mod_cesam`.

alecianl.f : Calcul des accélérations radiatives, routine PRIVATE du module `mod_evol`.

arb_rom.f : Transforme la notation arabe en romaine, fonction PUBLIC du module `mod_numerique`.

ascii.f : Création d'un fichier de sortie ASCII personnalisé, routine PRIVATE du module `mod_cesam`.

atm.f : Routine générique de restitution de l'atmosphère, routine PUBLIC du module `mod_atm`.

base_chim.f : Routine subordonnée de `diffus.f90`, formation de la base pour la diffusion des éléments chimiques.

base_rota.f : Routine subordonnée de `evol.f90` formation de la base continue non dérivable pour la rotation.

boite.f : Dessine une boite de dimensions $\pm dx \times dy$ autour du point de coordonnées (x, y), routine PUBLIC du module `mod_numerique`.

box.f : Dessine une boite asymétrique autour du point de coordonnées (x, y), routine PUBLIC du module `mod_numerique`.

bspldd n.f : Interpolation 1D avec une spline polynômiale + dérivées, routine PUBLIC du module `mod_numerique`.

bspldn.f : Interpolation 1D avec une spline polynômiale, routine PUBLIC du module `mod_numerique`.

bsp_dis.f : Interpolation 1D avec discontinuités avec une spline polynômiale, routine PUBLIC du module `mod_numerique`.

bval0.f : Calcul les B-splines normalisées en un point de leur domaine, routine PRIVATE du module `mod_numerique`.

bval1.f : Calcul les B-splines normalisées et de leurs dérivées 1-ières en un point de leur domaine, routine PUBLIC du module `mod_numerique`.

bvald.f : Calcul les B-splines normalisées et de leurs dérivées en un point de leur domaine, routine PUBLIC du module mod_numerique.

cesam.f : Gestion de l'évolution, constitue en fait le programme principal, routine PUBLIC du module mod_cesam.

cesam2k-dbg.f : Programme principal pour debug.

cesam2k.f : Programme principal pour exploitation.

cesamT-dbg.f : Programme principal pour debug de tests.

cesamT.f : Programme principal pour exploitation de tests.

chim_gram.f : Transforme les abondances par mole en abondances par gramme, routine PUBLIC du module mod_variables.

coeff_rota.f : Routine générique PRIVATE du module mod_evol calcul des coefficients pour la rotation.

coeff_rota3.f : Routine PRIVATE du module mod_evol calcul des coefficients pour la rotation avec le formalisme de Talon& Zahn 1997, Krot=3.

coeff_rota4.f : Routine PRIVATE du module mod_evol calcul des coefficients pour la rotation avec le formalisme de Mathis& Zahn 2004, Krot=4.

coeff_vth.f : Routine PRIVATE du module mod_etat, détermination des coefficients pour l'approximation vth de variables thermodynamiques.

coll.f : Détermination des abscisses des points de collocation, routine PUBLIC du module mod_numerique.

col_atm.f : Résolution des équations de l'atmosphère, routine PRIVATE du module mod_atm.

col_qs.f : Résolution des équations de l'équilibre quasi-statique, routine PRIVATE du module mod_static.

collision.f : Calcul des intégrales de collision, routine PRIVATE du module mod_evol.

colpnt.f : Initialisation des abscisses des points de collocation, routine PRIVATE du module mod_numerique.

conv.f : Routine générique du calcul de la convection, routine PUBLIC du module mod_conv.

conv_a0.f : Calcul de la convection MLT avec longueur de mélange nulle aux limites ZR/ZC, routine PRIVATE du module mod_conv.

conv_cgm_reza.f : Calcul de la convection selon Canuto & Mazitelli avec la modification CGM, routine PRIVATE du module mod_conv.

conv_cm.f : Calcul de la convection selon Canuto & Mazitelli, routine PRIVATE du module mod_conv.

conv_cm_reza.f : Calcul de la convection selon Canuto & Mazitelli avec la modification de R.Samadi, routine PRIVATE du module mod_conv.

conv_cml.f : Calcul de la convection selon Canuto & Mazitelli avec la longueur de mélange égale à la plus courte distance des bords de la zone convective, routine PRIVATE du module mod_conv.

conv_jmj.f : Calcul de la convection MLT, routine PRIVATE du module mod_conv.

coulomb.f : Calcul du logarithme de Coulomb, routine PRIVATE du module mod_evol.

ctes_85.f : Initialisation de constantes physiques, routine PRIVATE du module mod_donnees.

ctes_94.f : Initialisation de constantes physiques, routine PRIVATE du module mod_donnees.

ctes_94m.f : Initialisation de constantes physiques, routine PRIVATE du module mod_donnees.

delete_doubles : Elimination des éléments identiques dans un tableau ordonné. Routine PUBLIC du module mod_numerique.

des.f : Routine générique du dessin au cours de l'évolution, routine PRIVATE du module mod_cesam.

des_m.f : Dessin en masse au cours de l'évolution, routine PRIVATE du module mod_cesam.

des_r.f : Dessin en rayon au cours de l'évolution, routine PRIVATE du module mod_cesam.

df_rotx.f : transformation des dérivées partielles $(P, T, X) \rightarrow (\rho, T, X)$, routine PUBLIC du moduleмod_etat.

dgrad.f : Calcul de la différence des gradients, routine PUBLIC du module mod_variables.

difdiv.f : Calcul des différences divisées, routine PRIVATE du module mod_numerique.

diffm.f : Routine générique du calcul des coefficients de diffusion microscopique, routine PRIVATE du module mod_evol.

diffm_br.f : Calcul des coefficients de diffusion microscopique selon Burgers, routine PRIVATE du module mod_evol.

diffm_mp.f : Calcul des coefficients de diffusion microscopique selon Michaud & Proffit, routine PRIVATE du module mod_evol.

difft.f : Routine générique du calcul des coefficients de diffusion turbulente, routine PRIVATE du module mod_evol.

difft_gab.f : Routine PRIVATE du module mod_evol, formation du coefficient de diffusion turbulente, suivant une idée de M.Gabriel, on évite la sédimentation de l'hélium en mélangeant si T < Tlim=1.d6.

difft_nu.f : Calcul des coefficients de diffusion turbulente et radiative, routine PRIVATE du module mod_evol.

difft_smc.f : Calcul des coefficients de diffusion turbulente en tenant compte de la semi-convection. Routine PRIVATE du module mod_evol.

difft_sun.f : Routine PRIVATE du module mod_evol, formation du coefficient de diffusion turbulente, suivant une idée de M.Gabriel.

diffus.f : Gestion des équations de diffusion des espèces chimiques et du moment cinétique, routine PRIVATE du module mod_evol.

diffw.f : Routine générique du calcul des coefficients de diffusion du moment cinétique, routine PRIVATE du module mod_evol.

diffw_mpz.f : Calcul des coefficients de diffusion du moment cinétique, selon Mathis, Palacios & Zahn Mathis et al. (2004), routine PRIVATE du module mod_evol.

diffw_p03.f : Routine PRIVATE du module mod_evol, calcul des coefficients de diffusion turbulente pour la rotation Dh, Dv, Deff, suivant Palacios& al 2003.

dnunl.f : Calcul approximatif de la grande séparation, routine PRIVATE du module mod_cesam.

ecrit_ascii : Ecriture des variables de la rotation et des coefficients pour dessins, routine PUBLIC du module mod_evol.

ecrit_rota : Formation et écriture des fichiers ASCII, routine subordonnée de `cesam.f90`.

edding.f : Loi $T(\tau)$ d'Eddington, routine PRIVATE du module `mod_atm`.

eq_atm.f : Calcul des coefficients des équations de restitution de l'atmosphère, routine PRIVATE du module `mod_atm`.

eq_diff_chim.f : Calcul des coefficients des équations de diffusion de la composition chimique, routine PRIVATE du module `mod_evol`.

eq_diff_rota3.f : Calcul des coefficients des équations de diffusion du moment cinétique, routine PRIVATE du module `mod_evol`.

eq_diff_rota4.f : Calcul des coefficients des équations de diffusion du moment cinétique, routine PRIVATE du module `mod_evol`.

eq_ini_rota4.f : Calcul des coefficients des équations différentielles pour l'initialisation de la diffusion du moment cinétique suivant le formalisme de Mathis & Zahn (2004), routine PRIVATE du module `mod_evol`.

etat.f : Routine générique du calcul de l'équation d'état, routine PUBLIC du module `mod_etat`.

etat_ceff.f : Equation d'état CEFF, routine PRIVATE du module `mod_etat`.

etat_eff.f : Equation d'état EFF, routine PRIVATE du module `mod_etat`.

etat_gongl.f : Equation d'état GONG1, routine PRIVATE du module `mod_etat`.

etat_gong2.f : Equation d'état GONG2, routine PRIVATE du module `mod_etat`.

etat_mhd.f : Equation d'état MHD, routine PRIVATE du module `mod_etat`.

etat_opal.f : Equation d'état OPAL, routine PRIVATE du module `mod_etat`.

etat_opalX.f : Equation d'état OPAL utilisable pour $Z > 0.1$, routine PRIVATE du module `mod_etat`, *cf.* §7.29.1 (Page 178).

etat_opalZ.f : Equation d'état OPAL utilisable pour $Z > 0.1$, routine PRIVATE du module `mod_etat`, *cf.* §7.29.1 (Page 178).

evol.f : Gestion de l'évolution temporelle de la composition chimique, routine PUBLIC du module `mod_evol`.

f_rad.f : Routine générique du calcul des acclérations radiatives, routine PRIVATE du module `mod_evol`.

fcmax : Détermine la nécessité d'utiliser le nombre maximum de couche lors du calcul des derniers modèles. Routine subordonnée de `resout` du module `mod_static`.

fermi_dirac.f : Intégrales de Fermi-Dirac, routine PUBLIC du module `mod_numerique`.

from_alecian.f : package de calcul des accélérations radiatives Alécian2, routine PRIVATE du module `mod_bp_for_alecian`.

gauss_band.f : Résolution d'un système linéaire bande, routine PUBLIC du module `mod_numerique`.

hopf.f : Loi $T(\tau)$ de Hopf, routine PRIVATE du module `mod_atm`.

horner.f : Algorithme de Horner , routine PRIVATE du module `mod_numerique`.

hsra.f : Loi $T(\tau)$ HSRA, routine PRIVATE du module `mod_atm`.

iben.f : Calcul de l'énergie de contraction selon Iben, routine PRIVATE du module mod_nuc.

ini_ctes.f : Routine générique d'initialisation des constantes physiques, routine PUBLIC du module mod_donnees.

initialise_rota.f : Initialisation des variables de la diffusion du moment cinétique, routine PRIVATE du module mod_evol.

initialise_rota4.f : Initialisation des variables de la diffusion du moment cinétique, suivant le formalisme de Mathis & Zahn (2004), routine PRIVATE du module mod_evol.

integrales.f : Calcul de la valeur moyenne de la vitesse angulaire et de la variation temporelle du moment cinétique dans les zones convectives suivant le formalisme de Talon et al. (1997), routine PRIVATE du module mod_evol.

inter.f : Interpolation inverse des variables, routine PUBLIC du module mod_variables.

inter_atm.f : Interpolation inverse pour l'atmosphère, routine PUBLIC du module mod_exploit.

intgauss.f : Initialisation des poids et des abscisses pour l'intégration de Gauss, routine PUBLIC du module mod_numerique.

journal : Description des aménagements dans Cesam2k20.

jacob_rota.f : Calcul des premiers et seconds membres du jacobien pour la résolution du système différentiel non linéaire de la diffusion du moment cinétique, routine PRIVATE du module mod_evol.

k5750.f : Loi $T(\tau)$ de Kurucz, $T_{\rm eff} = 5750$K, routine PRIVATE du module mod_atm.

k5777.f : Loi $T(\tau)$ de Kurucz, $T_{\rm eff} = 5777$K, routine PRIVATE du module mod_atm.

kappa_cond.f : opacités conductives, routine PRIVATE du module mod_opa.

left_right.f : Calcul des n fonctions et de leurs dérivées premières de part et d'autre d'un point d'une interpolation B-Spline, routine PUBLIC du module mod_numerique.

lim_atm.f : Gestion de la restitution de l'atmosphère, routine PRIVATE du module mod_atm.

lim_gongl.f : Atmosphère monocouche, cas GONGl, routine PRIVATE du module mod_atm.

lim_taul.f : Atmosphère monocouche, routine PRIVATE du module mod_atm.

lim_zc.f : Gestion des limites ZR/ZC, routine PRIVATE du module mod_static.

linf.f : Recherche de l'encadrement d'une valeur, routine PUBLIC du module mod_numerique.

list.f : Listing des modèles, routine PRIVATE du module mod_cesam.

list_cesam : Liste des modules à compiler.

lit_binaire.f : Lecture d'un fichier binaire de type mon_modele_B.*.

lit_hr.f : Lecture des fichiers de type mon_modele.HR, routine PUBLIC du module mod_exploit.

lit_nl.f : Lecture des NAMELISTs, routine PUBLIC du module mod_donnees.

lmix.f : Fonction PRIVATE du module mod_evol détermine si on setrouve dans une zone de mélange pour la composition chimique

marcs.f : Lois $T(\tau)$ de MARCS, implantées par B.Pichon, routine PRIVATE du module mod_atm.

matinv.f : Inversion de matrice, routine PUBLIC du module mod_numerique.

max_local.f : Maxima locaux de plusieurs tables, routine PUBLIC du module mod_numerique.

min_max.f : Maximum/minimum d'une table, routine PUBLIC du module mod_numerique.

min_max_cond.f : Routine PUBLIC du module mod_exploit déterxmination de xmin et xmax sous condition pour des plots.

mod_atm.f : module regroupant les variables et les routines de résolution des équations différentielles pour la restitution de l'atmosphère.

mod_bp_for_alecian.f : module regroupant les routines de calcul de l'accélération radiative selon G.Alécian.

mod_cesam.f : module contenant ce qui constitue, de fait, le programme principal *i.e.* la routine cesam et les routines de gestion.

mod_conv.f : module regroupant les routines de convection.

mod_donnees.f : module contenant les données du modèle, les constantes physiques et leurs routines d'initialisation.

mod_etat.f : module regroupant les équations d'état.

mod_evol.f : module regroupant les routines relatives à l'évolution de la composition chimique.

mod_exploit.f : module regroupant des routines et programmes pour l'exploitation des modèles.

mod_kind.f : module regroupant les définitions de types de variables.

mod_nuc.f : module regroupant les routines des réactions nucléaires et d'initialisation de la mixture.

mod_numerique.f : module regroupant les outils numériques et informatiques.

mod_opa.f : module regroupant les routines d'opacité.

mod_static.f : module regroupant les variables et les routines de résolution des équations différentielles pour l'équilibre quasi-statique.

mod_variables.f : module regroupant les variables et quelques fonction de calcul de quantités dérivées des variables.

modif_mix.f : Routine permettant la personnalisation d'une mixture, routine PRIVATE du module mod_nuc.

neville.f : Algorithme de Neville, routine PUBLIC du module mod_numerique.

newspl.f : Changement de base de spline, routine PUBLIC du module mod_numerique.

newspl_gal.f : Changement de base de spline utilisant la formulation intégrale de Galerkin, routine PUBLIC du module mod_numerique.

newton.f : formule d'interpolation de Newton, routine PUBLIC du module mod_numerique.

noedif.f : Création de la base pour la résolution des équations différentielles selon de Boor, routine PUBLIC du module mod_numerique.

noein.f : Création de la base d'interpolation, routine du module mod_numerique.

noeu_dis.f : Création du vecteur nodal pour interpolation avec discontinuités, routine PRIVATE du module mod_numerique.

noeud.f : Création d'un vecteur nodal à partir d'un vecteur de mutiplicité, routine PUBLIC du module mod_numerique.

nuc.f : Routine générique de calcul des réactions nucléaires, routine PUBLIC du module mod_nuc.

opa.f : Routine générique du calcul d'opacité, routine PUBLIC du module mod_opa.

opa_compton.f : Routine PUBLIC du module mod_opa, Opacités free-free Compton utilisées pour T > 7d7K 7Kev.

opa_gong.f : opacité de GONG, routine PRIVATE du module mod_opa.

opa_houdek9.f : opacité OPAL + Alexander, interpolation de Houdek, routine PRIVATE du module mod_opa.

opa_int_zsx.f : opacité OPAL, routine PRIVATE du module mod_opa.

opa_opal2.f : opacité OPAL permettant de dépasser $Z > 0.1$, routine PRIVATE du module mod_opa, *cf.* §7.53 (Page 211).

opa_opalCO.f : opacité OPAL permettant de dépasser $Z > 0.1$, routine PRIVATE du module mod_opa, *cf.* §7.53 (Page 211).

opa_yveline.f : opacité OPAL + Alexander, interpolation de Yveline, routine PRIVATE du module mod_opa.

opa_yveline_lisse.f : opacité OPAL + Alexander, raccord d'Yveline, routine PRIVATE du module mod_opa.

osc_adia.f : formation du fichier d'oscillations adiabatiques, routine PRIVATE du module mod_cesam.

osc_invers.f : formation du fichier ASCII pour inversions, routine PRIVATE du module mod_cesam.

osc_nadia.f : formation du fichier d'oscillations non-adiabatiques, routine PRIVATE du module mod_cesam.

output.f : Routine générique de création des fichiers de sortie en ASCII, routine PRIVATE du module mod_cesam.

pause.f : pause avec commentaire, routine PUBLIC du module mod_numerique.

pertm.f : Routine générique du calcul de la perte de masse, routine PRIVATE du module mod_static.

pertm_ext.f : perte de masse linéaire, routine PRIVATE du module mod_static.

pertm_msol.f : perte de masse linéaire limitée, routine PRIVATE du module mod_static.

pertm_tot.f : perte de masse linéaire + $E = mc^2$, routine PRIVATE du module mod_static.

pertm_waldron.f : perte de masse selon Waldron, routine PRIVATE du module mod_static.

pertw.f : Routine générique de perte de moment cinétique, routine PRIVATE du module mod_evol.

pertw_loc.f : perte locale de moment cinétique, routine PRIVATE du module mod_evol.

pertw_ptm.f : perte de moment cinétique associée à une perte de masse, routine PRIVATE du module mod_evol.

pertw_sch.f : perte de moment cinétique selon Schumanich, routine PRIVATE du module mod_evol.

pgplot_factice.f : Packtage de routines émulant le logiciel de dessin PGPLOT, extérieur au module mod_cesam.

planetoides.f : Routine PUBLIC du module mod_nuc modifie les dérivées temporelles de la composition chimique calcule le moment cinétique par unité de masse apporté par les planétoïdes.

plot_rota.f : Routine subordonnée de ecrit_rota.f90 dessins des variables de la rotation.

polyder.f : Algorithme de Horner, routine PUBLIC du module mod_numerique.

pp1.f : Cycle PP simplifié, routine PRIVATE du module mod_chim.

pp3.f : Cycle PP, routine PRIVATE du module mod_chim.

ppcno10.f : Cycles PP + CNO 10 éléments, routine PRIVATE du module mod_chim.

ppcno10BeBFe.f : Cycles PP + CNO 10 éléments + Fe, routine PRIVATE du module mod_nuc.

ppcno10Fe.f : Cycles PP + CNO 10 éléments + Fe, routine PRIVATE du module mod_nuc.

ppcno10K.f : Cycles PP + CNO 10 éléments + K, routine PRIVATE du module mod_nuc.

ppcno11.f : Cycles PP + CNO 11 éléments, routine PRIVATE du module mod_nuc.

ppcno12.f : Cycles PP + CNO 12 éléments, routine PRIVATE du module mod_nuc.

ppcno12Be.f : Cycles PP + CNO 12 éléments + Be, routine PRIVATE du module mod_nuc.

ppcno12BeBFe.f : Cycles PP + CNO 12 éléments + Be, routine PRIVATE du module mod_nuc.

ppcno12Li.f : Cycles PP + CNO 12 éléments + Li, routine PRIVATE du module mod_nuc.

ppcno3a12Ne.f : Cycles PP + CNO + 3α 12 éléments, routine PRIVATE du module mod_nuc.

ppcno3a9.f : Cycles PP + CNO + 3α 9 éléments, routine PRIVATE du module mod_nuc.

ppcno3aco.f : Cycles PP + CNO + 3α + C + O, routine PRIVATE du module mod_nuc.

ppcno9.f : Cycles PP + CNO 9 éléments, routine PRIVATE du module mod_nuc.

ppcno9Fe.f : Cycles PP + CNO 9 éléments, routine PRIVATE du module mod_nuc.

print_ctes.f : Ecriture des constantes, routine PUBLIC du module mod_donnees.

read_ascii.f : Lecture du fichier ASCII, routine du module mod_exploit.

resout.f : Gestion de la résolution de l'équilibre quasi-statique + évolution temporelle, routine du module mod_static.

resout_chim.f : Routine PUBLIC du module mod_evol résolution par éléments finis Galerkin du système d'équa. diff. ord. non linéaires de la rotation par itération Newton-Raphson.

resout_rota.f : Routine générique PRIVATE du module mod_evol, résolution du système des équations de la diffusion du moment cinétique.

rk_imps.f : Algorithme Runge-Kutta implicite, routine PRIVATE du module mod_evol.

roger.f : Lois $T(\tau)$ Kurucz, routine PRIVATE du module mod_atm.

rq_reac.f : Calcul des énergies des réactions nucléaires, routine PRIVATE du module mod_nuc.

saha.f : Equation de Saha, routine PUBLIC du module `mod_etat`.

schu58_n.f : Algorithme de Schumaker, routine PRIVATE du module `mod_numerique`.

shell.f : Routine de tri, routine PUBLIC du module `mod_numerique`.

static.f : Routine générique du calcul des coefficients de l'équilibre quasi-statique, routine PRIVATE du module `mod_static`.

static_m.f : Calcul des coefficients de l'équilibre quasi-statique, cas lagrangien, routine PRIVATE du module `mod_quasi_static`.

static_r.f : Calcul des coefficients de l'équilibre quasi-statique, cas eulérien, routine PRIVATE du module `mod_quasi_static`.

sum_n.f : Intégration d'une spline, routine PUBLIC du module `mod_numerique`.

tabul_nuc.f : tabulation des réactions thermonucléaires, routine PRIVATE du module `mod_nuc`.

tab_vth.f : tabulation de variables thermodynamiques, routine PUBLIC du module `mod_evol`.

taueff.f : pour une loi $T(\tau)$, détermination de l'épaisseur optique où $T = T_{\rm eff}$, routine PRIVATE du module `mod_atm`.

taux_nuc.f : taux des réactions thermonucléaires, routine PRIVATE du module `mod_nuc`.

tdetau.f : Routine générique de loi $T(\tau)$, routine PUBLIC du module `mod_atm`.

thermo.f : Calcul des variables thermodynamiques pour l'équilibre quasi-statique, routine PUBLIC du module `mod_static`.

thermo_atm.f : Calcul des variables thermodynamiques pour l'atmosphère, routine PUBLIC du module `mod_atm`.

trho.f : Interpolation en T et ρ pour lois $T(\tau)$ de Kurucz, routine PRIVATE du module `mod_atm`.

update.f : passage des variables du temps t+dt au temps t, fonction PRIVATE du module `mod_static`.

vent.f : Gestion de la composition chimique du vent, routine PRIVATE du module `mod_nuc`.

write_nl.f : Ecriture des NAMELISTs, routine PRIVATE du module `mod_cesam`.

z14xcotrin21.f : Package d'interpolation d'opacité, extérieur au module `mod_opa`.

zoning.f : Répartition équidistante d'une fonction croissante, routine PUBLIC du module `mod_numerique`.

E.2 Sous-directory EXPLOIT

2d-2.pms : Fichier ASCII d'initialisation de pré-séquence principale, on obtient une température centrale de l'ordre de 100 000K.

5d-4.pms : Fichier ASCII d'initialisation de pré-séquence principale, on obtient une température centrale de l'ordre de 500 000K.

8d-5.pms : Fichier ASCII d'initialisation de pré-séquence principale, on obtient une température centrale de l'ordre de 1 000 000K.

blabla : Fichier ASCII d'affectation de la variable logique `blabla` permettant de détourner la plupart des commentaires permettant le suivi des calculs.

calib2k_pms.f : Programme de calibration, initialisation sur la PMS.

calib2k_zams.f : Programme de calibration, initialisation sur la ZAMS.

des2k_ZC.f : Programme de dessin des zones convectives.

des2k_abon.f : Programme de dessin des abondances.

des2k_abontc.f : Programme de dessin de X, Y et Z au centre, en fonction du temps.

des2k_abonts.f : Programme de dessin de X, Y et Z en surface, en fonction du temps.

des2k_bin.f : Programme de dessin des variables quasi-statiques en fonction de la masse ou du rayon à partir d'un fichier binaire.

des2k_diff_osc : Programme de dessin des différences de 2 fichiersd'oscillations de type `mon_modele-**.osc`, interpolation par `neville.f90`.

des2k_diff_spl : Programme de dessin des différences de 2 fichiersd'oscillations de type `mon_modele-**.osc`, interpolation par B-Spline.

des2k_grad : Programme de dessin des gradients.

des2k_hr.f : Programme de dessin de diagrammes HR.

des2k_opa.f : Programme de dessin des différences relatives de deux sources d'opacité, le long d'un modèle.

des2k_osc.f : Programme de dessin des principales variables d'un fichier d'oscillations de type `mon_modele-ad.osc`.

des2k_rot.f : Programme des dessins séparés des variables de la diffusion du moment cinétique à partir du fichier binaire optionnel créé dans la routine `coeff_rota.f90`.

des2k_rot_ext.f : Programme de dessin, en fonction du temps, de la vitesse de rotation externe en Km/s ou en radian/s à partir d'un fichier HR.

des2k_vaiss.f : Programme de dessin de la fréquence de **v**, à partir d'un fichier binaire de type `mon_modele_B.*`.

device : Exemple de ficheir de personnalisation des paramètres de dessin *"on line"*, *cf.* §3.15 (Page 30).

f037_2k.f : Formation d'un fichier pour le calcul des oscillations à partir d'un fichier binaire de type `mon_modèle_B.dat`.

fichier_vent.f : Programme de contruction, pour le vent, d'un fichier de composition chimique différente de celle des couches externes.

langue : Exemple de fichier de langue.

m010.zams : Fichier ASCII d'initialisation de séquence principale d'âge zéro homogène, pour $1M_\odot$.

m020.zams : Fichier ASCII d'initialisation de séquence principale d'âge zéro homogène, pour $2M_\odot$.

m050.zams : Fichier ASCII d'initialisation de séquence principale d'âge zéro homogène,pour $5M_\odot$.

mixture : Exemple de fichier de mixture initiale, différente de celles implémentées dans Cesam2k20, *cf.* §3.15.1 (Page 30).

modif_mix : Exemple de fichier de modification de mixture pour ***tous les modèles*** calculés dans l'environnement, *cf.* § 3.15.3 (Page 31).

mon_modele.don : Exemple de fichier de données.

mon_modele.modif_mix : Exemple de fichier de modification de mixture pour ***le*** modèle `mon_modele`.

mon_modele.rap_iso : Exemple de fichier de modification des rapports isotopiques pour ***le*** modèle `mon_modele`

mon_modele.vent : Exemple de fichier de vent pour ***le*** modèle `mon_modele`.

planet : Exemple de fichier permettant de définir les paramètres des chutes de planètoïdes, *cf.* § 6.7.2 (Page 87).

rap_iso : Exemple de fichier de modification des rapports isotopiques.

reglages : Exemple de fichier de réglage.

sortie_ascii : Exemple de fichier de personnalisation des sorties ASCII.

vent : Exemple de fichier de composition chimique du vent.

zoom : Exemple de fichier de personnalisation des dessins "on line".

E.3 Sous-directory TESTS

Certains de ces programmes de test ne sont peut être pas à jour de la toute dernière version de Cesam2k20.

plot_reac_nuc.f : Programme de dessin des taux des réactions nucléaires.

test_acc_rad.f : Programme de test des accélérations radiatives.

test_atm.f : Programme de test de l'intégration de l'atmosphère.

test_bsp_dis.f : Programme de test de l'interpolation par B-splines avec discontinuités.

test_cesam.f : Programme de test de Cesam2k20.

test_coef_diff.f : Programme de test du calcul des coefficients de diffusion microscopique.

test_coeff_rota.f : Programme de test du calcul des coefficients de la diffusion du moment cinétique.

test_collision.f : Programme de test des coefficients de collision.

test_colloc.f : Programme de test de la construction d'un vecteur nodal pour la collocation.

test_convection.f : Programme de test d'une routine de convection.

test_data_ceff.f : Programme de test des DATA de l'équation d'état CEFF.

test_data_eff.f : Programme de test des DATA de l'équation d'état EFF.

test_der_rotx.f : Programme de test de transformation de dérivation.

test_etat/2.f : Programmes de test d'Equation of State.

test_jacobien_reac_nuc.f : Programme de test du jacobien d'un réseau de réactions ucléaires.

test_lim_ZC.f : Programme de test de la localisation des zones convectives.

test_mu_mol.f : Programme de test de calcul du poids moléculaire moyen.

test_nl.f : Programme de test de lecture des NAMELISTs.

test_opa.f : Programme de test des opacités.

test_opa_opal2.f : Programme de test des opacités OPAL2.

test_opacite.f : Programme de test des opacités avec différentes mixtures.

test_read_osc.f : Programme de test de la lecture des fichiers ASCII d'oscillations.

test_rk_imps.f : Programme de test de la routine d'intégration des réactions du réseau nucléaire sans diffusion microscopique.

test_saha.f : Programme de test du calcul des taux d'ionisation.

test_tabul_reac.f : Programme de test de la tabulation des taux des réactions nucléaires.

test_tdetau.f : Programme de test d'un loi $T(\tau)$.

test_thermo.f : Programme de test du calcul des principales variables physiques de la structure interne.

test_thermo_atm.f : Programme de test du calcul des principales variables physiques de l'atmosphère.

E.4 Sous-directory SCRIPTS

makefile : MAKEFILE pour la création de l'exécutble `cesam2k.out` avec le compilateur `lf95`.

pgplot-5.2.0-2.i686.rpm : Source RPM de PGPLOT pour LINUX.

Les deux sous-directory SCRIPTS_CSH et SCRIPTS_BASH contennent des shell scripts respectivement en csh et bash, dont les fonctions sont les duivantes:

calib2k_pms : Calibration d'un modèle solaire incluant la PMS.

calib2k_zams : Calibration d'un modèle solaire initialisé sur la ZAMS homogène.

compile2k : Compilation d'une routine avec le compilateur `lf95`.

compile2k-dbg : Compilation d'une routine avec le compilateur `lf95` et options de debug.

compile2k-dbg_info : Compilation d'une routine avec le compilateur `lf95`, options de debug et constitution d'une bibliothèque suivant la liste de programmes du fichier `list`.

compile2k-dbg_list : Compilation d'une routine avec le compilateur `lf95`, options de debug et constitution d'une bibliothèque suivant la liste de programmes du fichier `list`.

compile2k_list : Compilation des routines de la liste du fichier `list` de l'environnement, avec le compilateur `lf95`.

evol2k_pms : Evolution d'un modèle incluant la PMS.

evol2k_zams : Evolution d'un modèle initialisé sur la ZAMS homogène.

exe2k : Exécution d'un programme avec le compilateur `lf95`.

exe2k-dbg : Exécution d'un programme avec le compilateur `lf95` et options de debug.

genere_cesam2k : Création de la bibliothèque et de l'exécutble `cesam2k.out` avec le compilateur `lf95`.

genere_cesam2k-dbg : Création de la bibliothèque et de l'exécutble `cesam2k-dbg.out` de debug avec le compilateur `lf95`

lib_del : Suppression de routines d'une bibliothèque.

lib_del_repl : suppression et remplacement de routines d'une bibliothèque.

lib_repl : Remplacement de routines d'une bibliothèque.

makefile : `MAKEFILE` pour la création de l'exécutble `cesam2k.out` avec le compilateur `lf95`.

rempl2k_mod : Remplacement d'un module et formation de l'exécutable pour exploitation.

rempl2k_mod-dbg : Remplacement d'un module et formation de l'exécutable pour debug.

E.5 Sous-directory SUN_STAR_DATA

*****.pms** : Fichiers ASCII d'initialisation de modèles de pré-séquence principale homogène.

Alrd*** : Opacités OPAL "Allard 96".

C95*** : Opacités OPAL "Cox 95".

COX*** : Opacités OPAL "Cox".

EOS*** : Données pour l'Equation of State OPAL.

GN*** : Opacités OPAL "Grevesse Noels".

GS*** : Données pour les opacités OPAL, *cf.* §7.85 (Page 233).

Gz*** : Données pour les opacités OPAL, *cf.* §7.85 (Page 233).

HOUDEK_17.03.06.tar.gz : Données pour les opacités Houdek, *cf.* §**??** (Page **??**)

IEOS*** : Données pour l'Equation of State OPAL.

W95*** : Opacités OPAL "Weiss", *cf.* §7.85 (Page 233).

ZFSinterppeos.f : Programme de création de tables d'Equation of State OPAL.

Z_interp_IEOS : Programme de création de tables d'Equation of State OPAL.

ascii2bin_opa : Programme de création de tables d'opacité OPAL en binaire.

datai_gr2.tar.gz : Données pour les accélérations radiatives Alécian2, *cf.* §7.8.2 (Page 160).

extract_opa_yveline.explik : Explications pour la création de fichiers d'opacité exploitables par `opa_yveline.f90`.

fesh*** : Données pour les tables de loi $T(\tau)$ de type `roger**`, *cf.* §**??** (Page **??**).

m*.zams.gz** : Fichiers ASCII d'initialisation de modèles de séquence principale d'âge zéro homogène.

mhdl.tab*** : Données pour l'Equation of State MHD, *cf.* §7.28.1 (Page 177).

opa_yveline.data.gz : Tables d'opacité Yveline. *cf.* §7.55 (Page 213)

opal* : Fichiers ASCII pour la création de tables d'Equation of StateOPAL.

opint_v9f.tar.gz : Package d'opacité Houdek, *cf.* §7.51 (Page 210),

peos* : Tables pour l'Equation of State OPAL.

phi_psi2* : Données pour le calcul des accélérations radiatives Alécianl, *cf.* §7.8.1 (Page 160).

Index

References

Adelberger, E. G., Austin, S. M., Bahcall, J. N., et al. 1998, Reviews of Modern Physics, 70, 1265

Aikawa, M., Arai, K., Arnould, M., Takahashi, K., & Utsunomiya, H. 2006, in American Institute of Physics Conference Series, Vol. 831, Frontiers in Nuclear Structure, Astrophysics, and Reactions, ed. S. V. Harissopulos, P. Demetriou, & R. Julin, 26–30

Alecian, G., Michaud, G., & Tully, J. 1993, Astrophysical Journal, 411, 882

Anders, E. & Grevesse, N. 1989, Geochimica et Cosmochimica Acta, 53, 197

Asplund, M., Amarsi, A. M., & Grevesse, N. 2021, Astronomy & Astrophysics, 653, A141

Asplund, M., Grevesse, N., & Sauval, A. J. 2005, Astronomical Society of the Pacific Conference Series, Vol. 336, The Solar Chemical Composition, ed. I. Barnes, Thomas G. & F. N. Bash, 25

Asplund, M., Grevesse, N., Sauval, A. J., & Scott, P. 2009, Annual Review of Astronomy & Astrophysics, 47, 481

Ball, W. H. 2021, Research Notes of the American Astronomical Society, 5, 7

Baturin, V. A., Ayukov, S. V., Gryaznov, V. K., et al. 2013, in Astronomical Society of the Pacific Conference Series, Vol. 479, Progress in Physics of the Sun and Stars: A New Era in Helio- and Asteroseismology, ed. H. Shibahashi & A. E. Lynas-Gray, 11

Böhm-Vitense, E. 1958, Zeitschrift für Astrophysik, 46, 108

Broggini, C., Bemmerer, D., Caciolli, A., & Trezzi, D. 2018, Progress in Particle and Nuclear Physics, 98, 55

Burgers, J. M. 1969, Flow Equations for Composite Gases

Canuto, V. M., Goldman, I., & Mazzitelli, I. 1996, ApJ, 473, 550

Canuto, V. M. & Mazzitelli, I. 1991, ApJ, 370, 295

Castro, M., Vauclair, S., & Richard, O. 2007, Astronomy & Astrophysics, 463, 755

Caughlan, G. R. & Fowler, W. A. 1988, Atomic Data and Nuclear Data Tables, 40, 283

Christensen-Dalsgaard, J. 1988, Astronomisk Institut, Aarhus Universitet

Clayton, D. D. 1968, Principles of stellar evolution and nucleosynthesis

Coc, A. & Vangioni, E. 2017, International Journal of Modern Physics E, 26, 1741002

Cox, J. P. & Giuli, R. T. 1968, Principles of stellar structure

Cyburt, R. H., Amthor, A. M., Ferguson, R., et al. 2010, Astrophysical Journal, Supplement Series, 189, 240

Daniel, F., Petitdemange, L., & Gissinger, C. 2023, Physical Review Fluids, 8, 123701

de Boor, C. 1978, A practical guide to splines

Deal, M., Goupil, M. J., Marques, J. P., Reese, D. R., & Lebreton, Y. 2020, Astronomy & Astrophysics, 633, A23

Eggleton, P. P. 1971, Monthly Notices of the Royal Astronomical Society, 151, 351

Eggleton, P. P. 1972, Monthly Notices of the Royal Astronomical Society, 156, 361

Eggleton, P. P., Faulkner, J., & Flannery, B. P. 1973, Astronomy & Astrophysics, 23, 325

Gabriel, M. 1997, Astronomy & Astrophysics, 327, 771

Grevesse, N. & Noels, A. 1993, Physica Scripta Volume T, 47, 133

Grevesse, N. & Sauval, A. J. 1998, Space Science Reviews, 85, 161

Hairer, E. & Wanner, G. 1996, Solving Ordinary Differential Equations II, 2nd edn., Springer Series in Computational Mathematics (Springer Berlin, Heidelberg)

Henrici, P. 1962, Discrete variable methods in ordinary differential equations

Henyey, L., Vardya, M. S., & Bodenheimer, P. 1965, Astrophysical Journal, 142, 841

Henyey, L. G., Lelevier, R., & Levee, R. D. 1959, Astrophysical Journal, 129, 2

Iben, I. J. 1965, Astrophysical Journal, 141, 993

Iben, I., J. 1975, Astrophysical Journal, 196, 525

Iglesias, C. A. & Rogers, F. J. 1996, Astrophysical Journal, 464, 943

Kawaler, S. D. 1988, Astrophysical Journal, 333, 236

Kippenhahn, R. & Weigert, A. 1991, stellar-structure and Evolution, A&A Library (Springer-Verlag)

Kippenhahn, R., Weigert, A., & Hofmeister, E. 1967, Methods in Computational Physics, 7, 129

Krishna Swamy, K. S. 1966, Astrophysical Journal, 145, 174

Lide, David R., e. 1994, CRC Handbook of Chemistry and Physics, 75th edn. (CRC Press)

Ludwig, H.-G., Freytag, B., & Steffen, M. 1999, Astronomy & Astrophysics, 346, 111

Maeder, A. & Meynet, G. 2004, Astronomy & Astrophysics, 422, 225

Magic, Z., Collet, R., Asplund, M., et al. 2013, Astronomy & Astrophysics, 557, A26

Marchouk, G. & Agochkov, V. 1985, Finite elements method, ed. M. Moscou

Mathis, S., Palacios, A., & Zahn, J. P. 2004, Astronomy & Astrophysics, 425, 243

Mathis, S., Prat, V., Amard, L., et al. 2018, Astronomy & Astrophysics, 620, A22

Mathis, S. & Zahn, J.-P. 2004, Astronomy & Astrophysics, 425, 229

Matt, S. P., Brun, A. S., Baraffe, I., Bouvier, J., & Chabrier, G. 2015, Astrophysical Journal Letters, 799, L23

Michaud, G. J. & Proffitt, C. R. 1993, in Astronomical Society of the Pacific Conference Series, Vol. 44, IAU Colloq. 138: Peculiar versus Normal Phenomena in A-type and Related Stars, ed. M. M. Dworetsky, F. Castelli, & R. Faraggiana, 439

Mihalas, D., Dappen, W., & Hummer, D. G. 1988, Astrophysical Journal, 331, 815

Mitler, H. E. 1977, Astrophysical Journal, 212, 513

Morel, P. 1997, Astronomy and Astrophysics Supplement Series, 124, 597

Morel, P., van't Veer, C., Provost, J., et al. 1994, Astronomy and Astrophysics, 286, 91

Palacios, A., Talon, S., Charbonnel, C., & Forestini, M. 2003, Astronomy & Astrophysics, 399, 603

Press, W. H., Teukolsky, S. A., Vetterling, W. T., et al. 1995, Numerical Recipes in C, The Art of Scientific Computing, 2nd edn. (Cambridge)

Rogers, F. J. & Iglesias, C. A. 1992, Astrophysical Journal, 401, 361

Rogers, F. J. & Nayfonov, A. 2002, ApJ, 576, 1064

Schatzman, E. & Praderie, F. 1990, Les Etoiles (Ed. InterEditions/Editions du CNRS)

Schumaker, L. 1981, Splines Functions: Basic Theory, ed. N.-Y. John Whiley

Serenelli, A. M., Basu, S., Ferguson, J. W., & Asplund, M. 2009, Astrophysical Journal Letters, 705, L123

Spruit, H. C. 2002, Astronomy & Astrophysics, 381, 923

Stoer, J. & Bulirsch, R. 1979, Introduction to Numerical Analysis (Springer-Verlag, Berlin)

Strittmatter, P. A., Faulkner, J., Robertson, J. W., & Faulkner, D. J. 1970, Astrophysical Journal, 161, 369

Talon, S., Zahn, J. P., Maeder, A., & Meynet, G. 1997, Astronomy & Astrophysics, 322, 209

Tanner, J. D., Basu, S., & Demarque, P. 2016, Astrophysical Journal Letters, 822, L17

Trampedach, R., Stein, R. F., Christensen-Dalsgaard, J., Nordlund, Å., & Asplund, M. 2014, Monthly Notices of the Royal Astronomical Society, 442, 805

Trenoguine, V. 1980, Analyse Fonctionnelle, ed. M. Moscou

Vernazza, J. E., Avrett, E. H., & Loeser, R. 1981, Astrophysical Journal, Supplement Series, 45, 635

Xu, Y., Takahashi, K., Goriely, S., et al. 2013, Nuclear Physics A, 918, 61

Zahn, J. P. 1991, Astronomy & Astrophysics, 252, 179