NSCool User's guide

Introduction

Dany Page

Instituto de Astronomía
Universidad Nacional Autónoma de México

NSCool

NSCool is a 1D (i.e., spherically symmetric) neutron star cooling code, written in fortran 77. Besides the cooling code, the package also contains a series of EOSs (equation of state) to build stars, a series of pre-built stars, and a TOV (Tolman- Oppenheimer-Volkoff) integrator to build stars from an EOS.

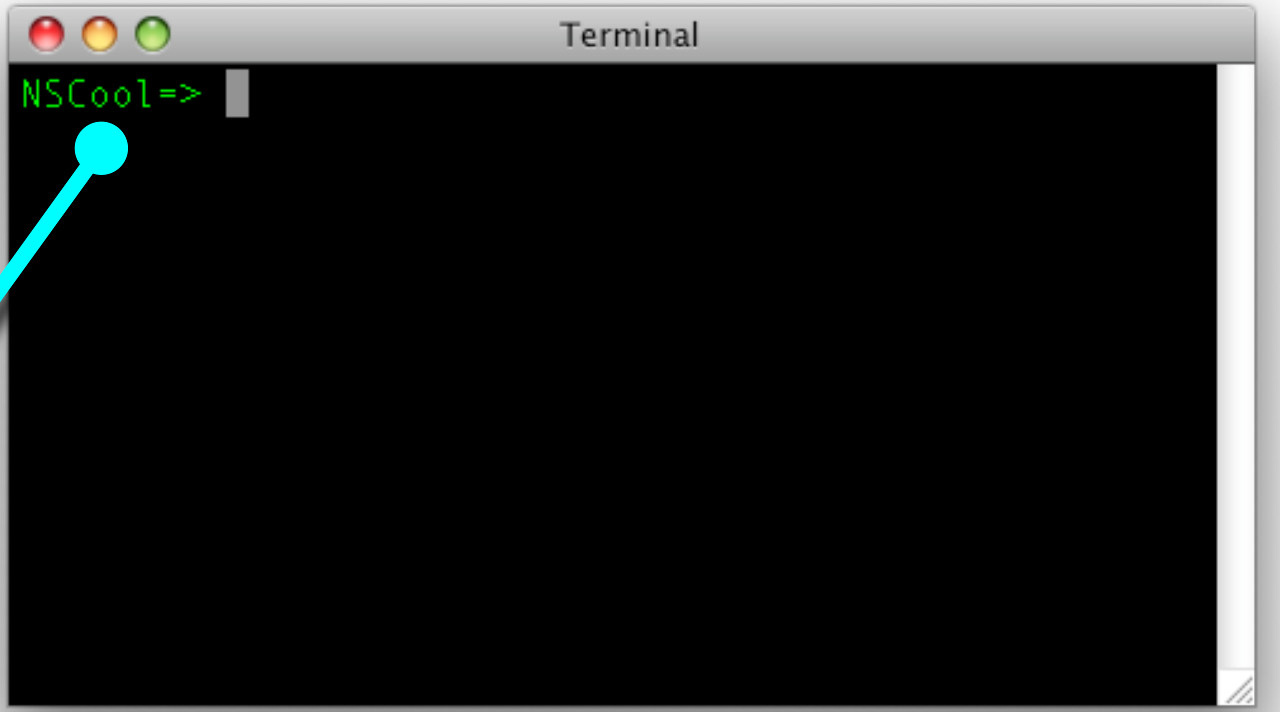
This code is extremely robust, very fast (the whole cooling of an isolated neutron star from birth to freezing takes less than a minute even on a lousy laptop), and highly modular (which makes it easy to add new subroutines for new processes without major risks of screwing it up !). It can also handle “strange stars”, which have a huge density discontinuity between the quark matter and the covering thin baryonic crust.

Once given a star to cool, with all the wanted/necessary physics set up (several configuration files are provided as examples), and an initial temperature profile, the code solves the heat transport and energy balance equations, in whole GR. The result is a time sequence of temperature profiles (and, in particular, a T_{eff} - age curve).

Several heating processes are included, and more can easily be incorporated. In particular it can evolve a star undergoing accretion with the resulting deep crustal heating, under a steady or time-variable accretion rate.

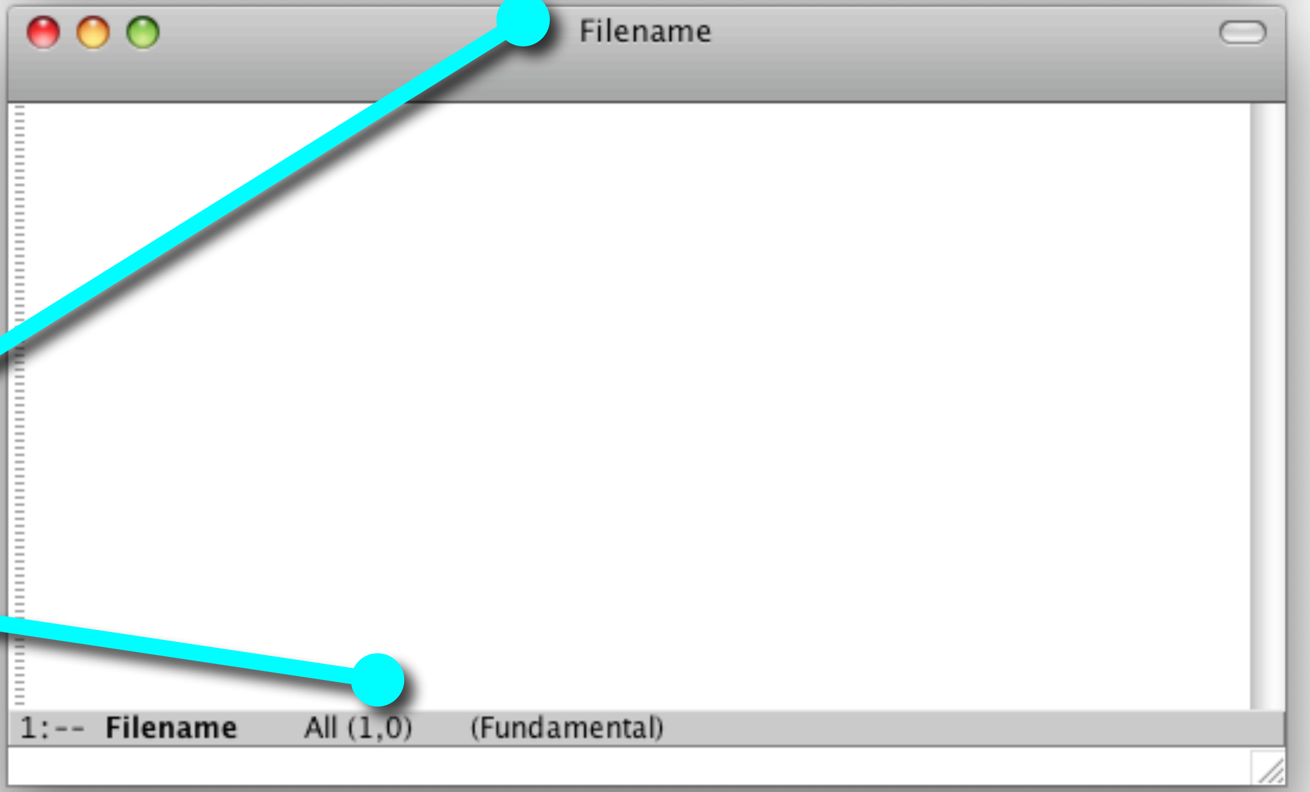
Conventions used in this User’s Guide

This is a terminal window:
- will show commands to run
  (I am using the bash shell)
- The prompt will show the
  working directory:



```
NSCool=>
```

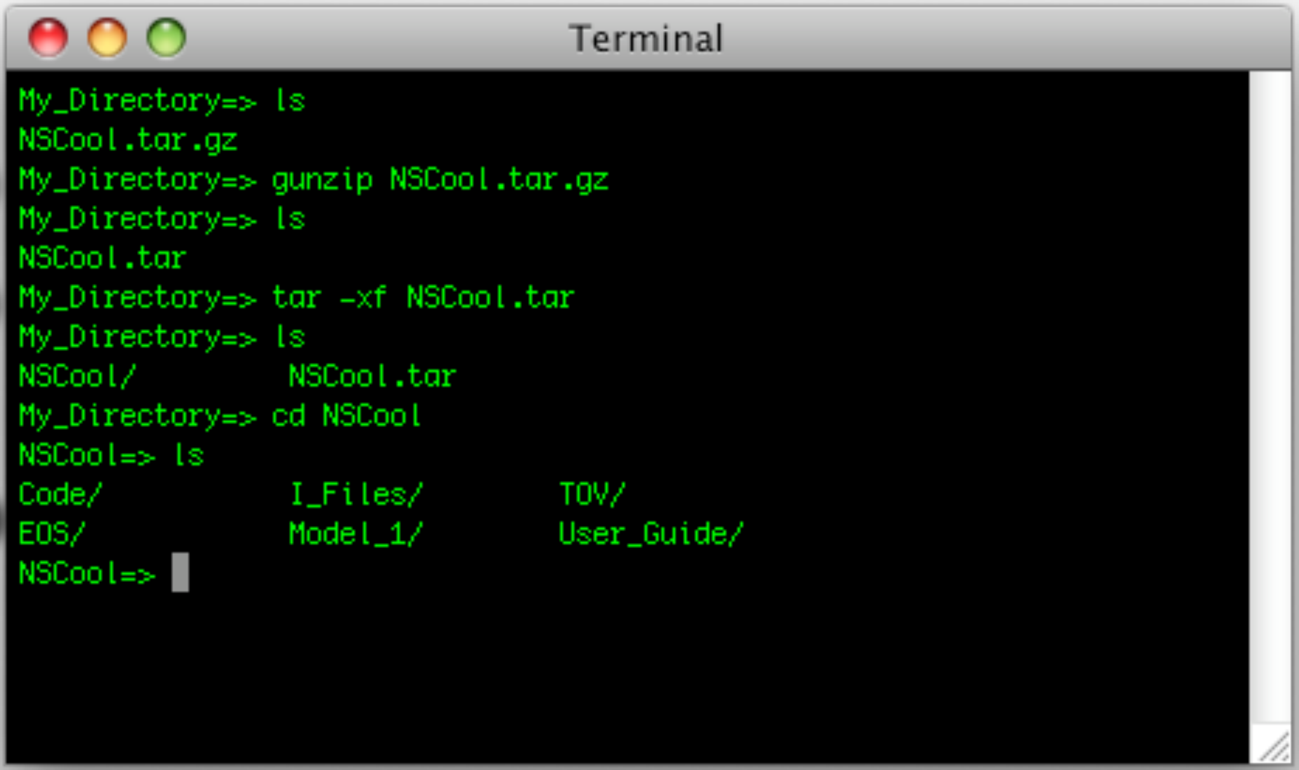
This is an editor window:
- will show file contents
  (I am using Aquamacs, i.e.,
  Emacs for Mac OSX)
- The title bar shows the file name:
- and this shows the line (and column)
  number in the file where the cursor is:



```
1:--  Filename     All (1,0)     (Fundamental)
```

Unpack the code

- Go to the directory where to install the code (“My_Directory” in the example) and copy the archive NSCool.tar.gz to this directory.
- Unpack the NSCool.tar.gz archive: will create a directory NSCool with several subdirectories:



```
My_Directory=> ls
NSCool.tar.gz
My_Directory=> gunzip NSCool.tar.gz
My_Directory=> ls
NSCool.tar
My_Directory=> tar -xf NSCool.tar
My_Directory=> ls
NSCool/         NSCool.tar
My_Directory=> cd NSCool
NSCool=> ls
Code/           I_Files/        TOV/
EOS/            Model_1/        User_Guide/
NSCool=> 
```

- Code contains the code.
- EOS contains EOSs.
- TOV contains the TOV solver and pre-built stars.
- I_Files contains samples of control files needed for running the code.
- Model_1 contains examples to run.
- User_Guide contains what it says.

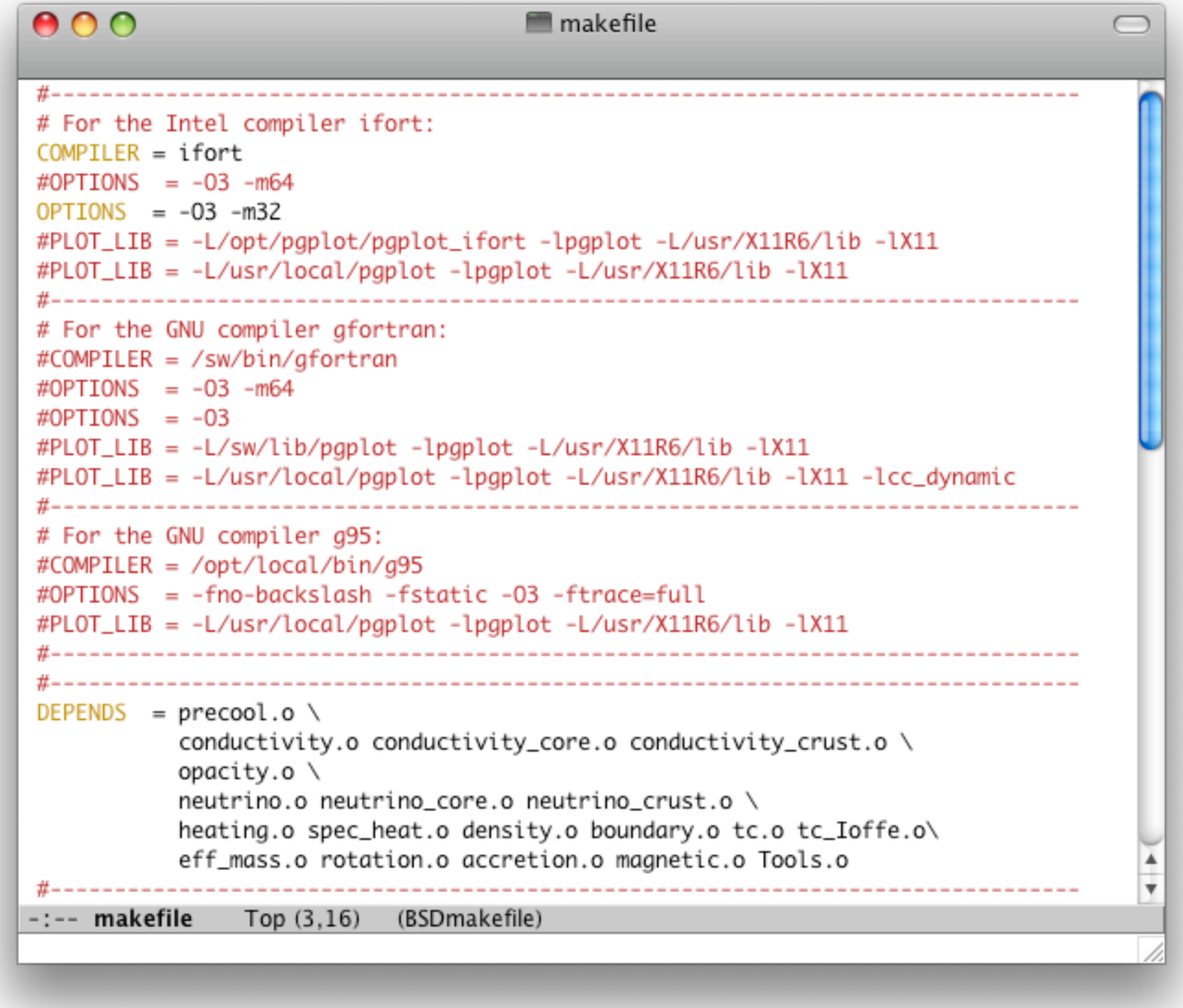
(There may be more directories, depending the version of the archive you have.)

Prepare the makefile for compilation

For what I have checked NSCool compiles OK with the GNU *g95* and *gfortran* compilers and with the Intel© *ifort*.

Edit the makefile (in the subdirectory Code) and define correspondingly the COMPILER and OPTIONS variables.
(If you have doubts about OPTIONS, try to just comment it out)

Save the makefile !



```
#------------------------------------------------------------------------
# For the Intel compiler ifort:
COMPILER = ifort
#OPTIONS  = -O3 -m64
OPTIONS  = -O3 -m32
#PLOT_LIB = -L/opt/pgplot/pgplot_ifort -lpgplot -L/usr/X11R6/lib -lX11
#PLOT_LIB = -L/usr/local/pgplot -lpgplot -L/usr/X11R6/lib -lX11
#------------------------------------------------------------------------
# For the GNU compiler gfortran:
#COMPILER = /sw/bin/gfortran
#OPTIONS  = -O3 -m64
#OPTIONS  = -O3
#PLOT_LIB = -L/sw/lib/pgplot -lpgplot -L/usr/X11R6/lib -lX11
#PLOT_LIB = -L/usr/local/pgplot -lpgplot -L/usr/X11R6/lib -lX11 -lcc_dynamic
#------------------------------------------------------------------------
# For the GNU compiler g95:
#COMPILER = /opt/local/bin/g95
#OPTIONS  = -fno-backslash -fstatic -O3 -ftrace=full
#PLOT_LIB = -L/usr/local/pgplot -lpgplot -L/usr/X11R6/lib -lX11
#------------------------------------------------------------------------
#------------------------------------------------------------------------
DEPENDS  = precool.o \
           conductivity.o conductivity_core.o conductivity_crust.o \
           opacity.o \
           neutrino.o neutrino_core.o neutrino_crust.o \
           heating.o spec_heat.o density.o boundary.o tc.o tc_Ioffe.o\
           eff_mass.o rotation.o accretion.o magnetic.o Tools.o
#------------------------------------------------------------------------
-:--  makefile      Top (3,16)     (BSDmakefile)
```

Compile NSCool ! ☺

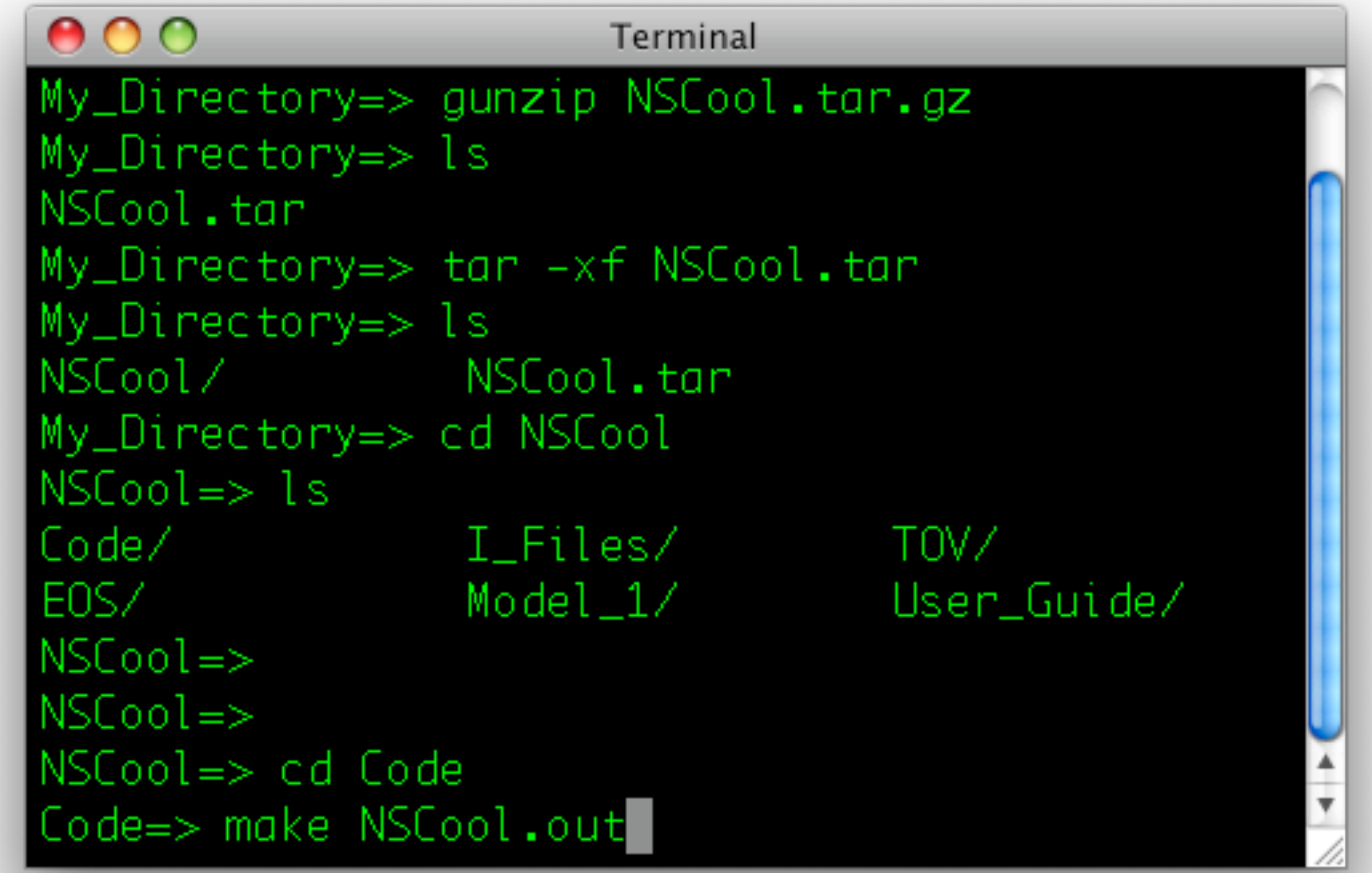
Just in case: there may be object files (as, e.g., NSCool.o) from a previous compilation in the directory Code. Delete them (with rm -f *.o) before compiling !

In subdirectory Code just type:

make NSCool.out <ENTER>

(and cross fingers)

If successful, this will create NSCool.out (which is also copied to the upper directory NSCool).



```
My_Directory=> gunzip NSCool.tar.gz
My_Directory=> ls
NSCool.tar
My_Directory=> tar -xf NSCool.tar
My_Directory=> ls
NSCool/          NSCool.tar
My_Directory=> cd NSCool
NSCool=> ls
Code/            I_Files/         TOV/
EOS/             Model_1/         User_Guide/
NSCool=>
NSCool=>
NSCool=> cd Code
Code=> make NSCool.out
```

If it does not work: you have a problem to solve !

Ask me, try to solve it by yourself, or
e-mail me: page@astro.unam.mx
(but this is not a support hot-line, there is no guarantee I will have time to respond in a useful manner. Which does not mean I will not do it ! Just no guarantee.)

A first trial run of *NSCool*

After compilation you have a copy of `NSCool.out` in the directory `NSCool`: you *must* run it from that directory !

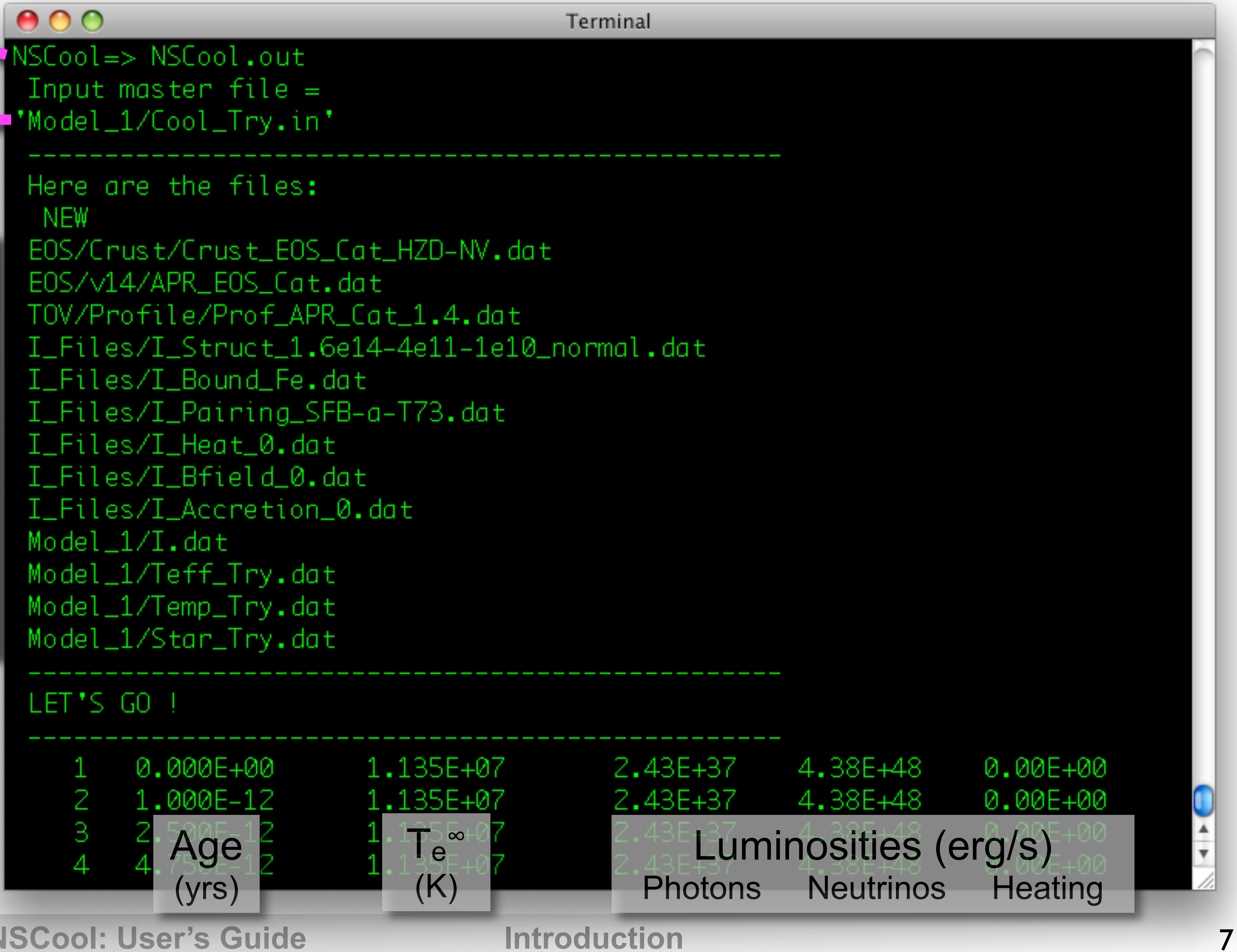
(so it can find all necessary files in the corresponding subdirectories).

1- Start `NSCool.out`

2- Type in the name of the "input "master file"

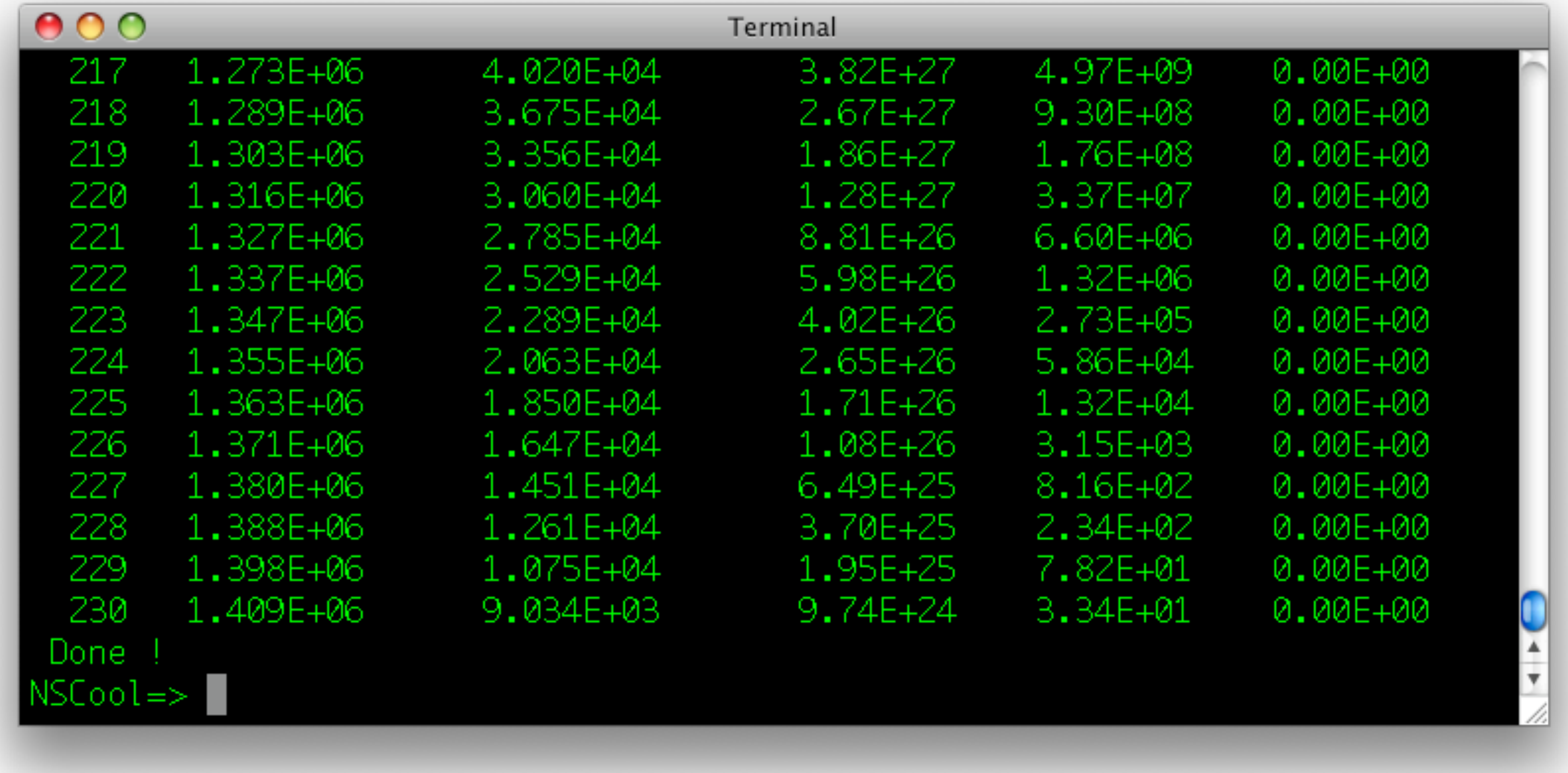
`NSCool` prints out the names of the files listed in the master file: these files define the model being calculated [more on this later]

☛ The print out is:



```
NSCool=> NSCool.out
 Input master file =
'Model_1/Cool_Try.in'
 -------------------------------------------------
 Here are the files:
  NEW
 EOS/Crust/Crust_EOS_Cat_HZD-NV.dat
 EOS/v14/APR_EOS_Cat.dat
 TOV/Profile/Prof_APR_Cat_1.4.dat
 I_Files/I_Struct_1.6e14-4e11-1e10_normal.dat
 I_Files/I_Bound_Fe.dat
 I_Files/I_Pairing_SFB-a-T73.dat
 I_Files/I_Heat_0.dat
 I_Files/I_Bfield_0.dat
 I_Files/I_Accretion_0.dat
 Model_1/I.dat
 Model_1/Teff_Try.dat
 Model_1/Temp_Try.dat
 Model_1/Star_Try.dat
 -------------------------------------------------
 LET'S GO !
 -------------------------------------------------
     1   0.000E+00      1.135E+07       2.43E+37   4.38E+48    0.00E+00
     2   1.000E-12      1.135E+07       2.43E+37   4.38E+48    0.00E+00
     3   2.500E-12      1.135E+07       2.43E+37   4.38E+48    0.00E+00
     4   4.750E-12      1.135E+07       2.43E+37   4.38E+48    0.00E+00
```

A first trial run of *NSCool*



```
  217   1.273E+06     4.020E+04      3.82E+27    4.97E+09    0.00E+00
  218   1.289E+06     3.675E+04      2.67E+27    9.30E+08    0.00E+00
  219   1.303E+06     3.356E+04      1.86E+27    1.76E+08    0.00E+00
  220   1.316E+06     3.060E+04      1.28E+27    3.37E+07    0.00E+00
  221   1.327E+06     2.785E+04      8.81E+26    6.60E+06    0.00E+00
  222   1.337E+06     2.529E+04      5.98E+26    1.32E+06    0.00E+00
  223   1.347E+06     2.289E+04      4.02E+26    2.73E+05    0.00E+00
  224   1.355E+06     2.063E+04      2.65E+26    5.86E+04    0.00E+00
  225   1.363E+06     1.850E+04      1.71E+26    1.32E+04    0.00E+00
  226   1.371E+06     1.647E+04      1.08E+26    3.15E+03    0.00E+00
  227   1.380E+06     1.451E+04      6.49E+25    8.16E+02    0.00E+00
  228   1.388E+06     1.261E+04      3.70E+25    2.34E+02    0.00E+00
  229   1.398E+06     1.075E+04      1.95E+25    7.82E+01    0.00E+00
  230   1.409E+06     9.034E+03      9.74E+24    3.34E+01    0.00E+00
 Done !
NSCool=>
```

CONGRATULATIONS: you have a working version of `NSCool` !

The Master and Control Files

For more details on the content of the “Input master file” (Cool_*.in)
and the contents of the many files listed in it, see:

```
NSCool_Guide_Control
```

(These files are of course essential
for doing anything beyond just running
the trial model)

What NSCool does

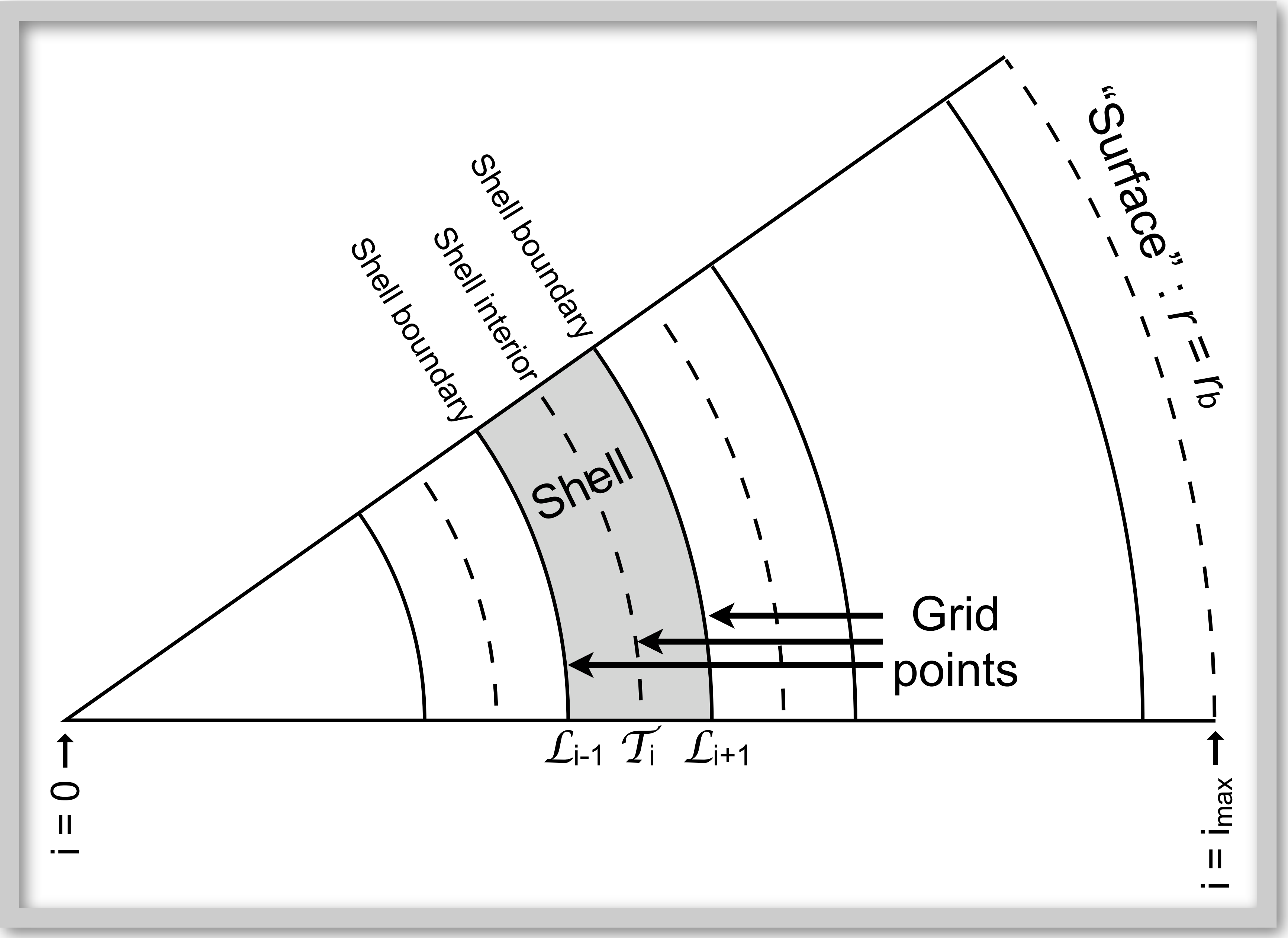
Given the initial conditions, NSCool solve the (fully GR version) of the equations of energy balance (i.e., conservation of energy) and energy transport. It uses as functions the internal temperature and the internal luminosity, actually the red-shifted ones:

$$\mathcal{T} \equiv e^{\Phi} T \quad \text{and} \quad \mathcal{L} \equiv e^{2\Phi} L$$

The star is discretized, at radii r_i, i=0, 1, 2, ...,i_{max} with $r_{i=0} = 0$ and $r_{i=imax} = r_b$ (the outer radius in the simulation)

$\mathcal{L}$ is defined at the boundaries between shells, i.e., *$i = 0, 2, 4, \ldots, i_{max}-1$*

$\mathcal{T}$ is defined inside the shells, i.e., *$i = 1, 3, 5, \ldots, i_{max}$*



The thermal evolution equations

The **energy balance** equation can be summarized as:

$$\frac{d\mathcal{T}}{dt} = F\left(Q_h, Q_\nu, C_v; \frac{d\mathcal{L}}{dr}\right)$$

which expresses that $\mathcal{T}$ changes with time because of the gradient of $\mathcal{L}$ and the presence of energy sources Q_h (heating rate), energy sinks Q_v (neutrino emissivity) and the capacity of energy storage C_v (specific heat).

The **energy transport** equation can be summarized as:

$$\mathcal{L} = G\left(\lambda, \frac{d\mathcal{T}}{dr}\right)$$

which expresses L in terms of the temperature gradient and the thermal conductivity λ.

Obviously $\mathcal{T}=T(r,t)$ and $\mathcal{L}=\mathcal{L}(r,t)$ so these are partial differential equations

These equations must be supplemented by boundary conditions:

- At the center of the star: $\mathcal{L}(r=0,t) \equiv 0$

  (no point source or sink of energy at the center of the star).
- At the surface of the star: $\mathcal{L} = 4\pi R_\infty^2 \sigma_{\rm SB} T_{e\,\infty}^4$

  (actually one uses something a little more sophisticated but physically equivalent: one glues an envelope model to the surface, which is then at $r_b < R$. This has many advantages, in particular it makes life easier).

Following the time evolution of $\mathcal{T}$ and $\mathcal{L}$

The time evolution of the $\mathcal{T}$ and $\mathcal{L}$ profiles goes step by step in time, e.g.,, from *t* to *t'=t+dt.*

Writing as $\mathcal{T}^{\text{old}}$ and $\mathcal{L}^{\text{old}}$ the profiles at time t and by just $\mathcal{T}$ and $\mathcal{L}$ at time t' one can try:

Explicit Method

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{dr}\right) \longrightarrow \mathcal{T} = \mathcal{T}^{\text{old}} + dt \cdot F\left(\mathcal{T}^{\text{old}}, \frac{d\mathcal{L}^{\text{old}}}{dr}\right)$$

$$\mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{dr}\right) \longrightarrow \mathcal{L} = G\left(\mathcal{T}^{\text{old}}, \frac{d\mathcal{T}^{\text{old}}}{dr}\right)$$

This looks *very simple* and is a *very bad idea*: it is numerically unstable !

Which means: unless you take ridiculously small time steps *dt* ("Courant dixit") your screen will fill up with a huge series of fearful "NAN".
(numerically: any error, and numerics are never exacts, will grow exponentially with time and blow up in your face)

The smart way: (not my idea, but due to L.G. Henyey. I first tried the above till somebody told me it was a stupid idea)

evaluate the r.h.s. using the new $\mathcal{T}$ and $\mathcal{L}$ instead of the previous values $\mathcal{T}^{\text{old}}$ and $\mathcal{L}^{\text{old}}$.

Implicit Method

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{dr}\right) \longrightarrow \mathcal{T} = \mathcal{T}^{\text{old}} + dt \cdot F\left(\mathcal{T}, \frac{d\mathcal{L}}{dr}\right)$$

$$\mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{dr}\right) \longrightarrow \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{dr}\right)$$

This is numerically stable but:

one has to extract $\mathcal{T}$ and $\mathcal{L}$ from it (and $\mathcal{T}$ is inside Q_h, Q_v, C_v and λ)

Iterating the $\mathcal{T}$ and $\mathcal{L}$ profiles

Solving the implicit form of the thermal evolution equations is done by iterations. Assuming we know the profiles of $\mathcal{T}$ and $\mathcal{L}$ at time *t*: $\mathcal{T}^{\mathrm{old}}$ and $\mathcal{L}^{\mathrm{old}}$ we will find the new $\mathcal{T}$ and $\mathcal{L}$ at time *t'=t+dt* by successive approximations

$$(\mathcal{T}_i^{(0)}, \mathcal{L}_i^{(0)}) \rightarrow (\mathcal{T}_i^{(1)}, \mathcal{L}_i^{(1)}) \rightarrow (\mathcal{T}_i^{(2)}, \mathcal{L}_i^{(2)}) \rightarrow (\mathcal{T}_i^{(3)}, \mathcal{L}_i^{(3)}) \rightarrow \ \ldots$$

[The notation ($\mathcal{T}_i$, $\mathcal{L}_i$) represents the array of $\mathcal{T}_i$'s (i=1, 3, 5, ...) and $\mathcal{L}_i^{(0)}$'s (i=0, 2, 4, ...) at the r_i's in the star]

We take a first guess ($\mathcal{T}_i^{(0)}$, $\mathcal{L}_i^{(0)}$) for the new ($\mathcal{T}_i$, $\mathcal{L}_i$), see how bad the equations are not solved, calculate corrections, plug back the new values ($\mathcal{T}_i^{(1)}$, $\mathcal{L}_i^{(1)}$) in the equations, see how bad the equations are not solved, calculate new corrections, a.s.o...

So, the *k*th to *(k+1)*th iteration, with corrections $\delta\mathcal{T}_i^{(k)}$ and $\delta\mathcal{L}_i^{(k)}$, looks like:

$$\mathcal{T}_i^{(k)} \rightarrow \mathcal{T}_i^{(k+1)} = \mathcal{T}_i^{(k)} + \delta\mathcal{T}_i^{(k)} \quad [i=1, 3, 5, \ldots]$$

$$\mathcal{L}_i^{(k)} \rightarrow \mathcal{L}_i^{(k+1)} = \mathcal{L}_i^{(k)} + \delta\mathcal{L}_i^{(k)} \quad [i=0, 2, 4, \ldots]$$

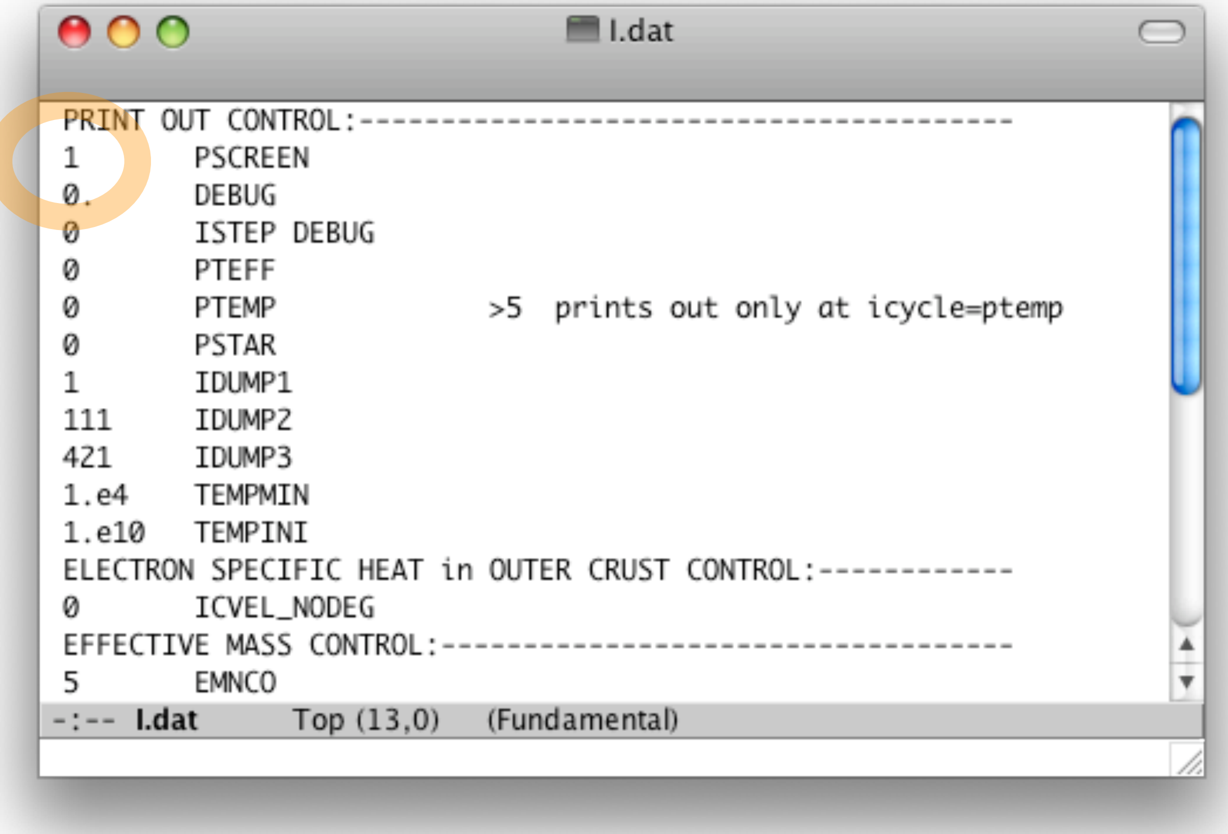
These iterations are stopped when corrections become small enough, i.e., when:

$$\mathrm{Max}_{i=1,3,5,\ldots}\left(\frac{\delta\mathcal{T}_i^{(k)}}{\mathcal{T}_i^{(k)}}\right) < \epsilon_T \quad \text{and} \quad \mathrm{Max}_{i=0,2,4,\ldots}\left(\frac{\delta\mathcal{L}_i^{(k)}}{\mathcal{L}_i^{(k)}}\right) < \epsilon_L$$

Accuracy of the order of 10^{-10} can be achieved in about 5 iterations !

Follow NSCool at work

To have NSCool print out more information on the screen while it is running edit the file `I.dat` (in the subdirectory `Model_1`) and change the value of the parameter `PSCREEN` ("print on screen") to 3 ("3", not "3.", and save the file).



```
PRINT OUT CONTROL:---------------------------------------
1       PSCREEN
0.      DEBUG
0       ISTEP DEBUG
0       PTEFF
0       PTEMP              >5  prints out only at icycle=ptemp
0       PSTAR
1       IDUMP1
111     IDUMP2
421     IDUMP3
1.e4    TEMPMIN
1.e10   TEMPINI
ELECTRON SPECIFIC HEAT in OUTER CRUST CONTROL:------------
0       ICVEL_NODEG
EFFECTIVE MASS CONTROL:----------------------------------
5       EMNCO
```

Now, rerun NSCool !

[Now, at each time step you have to press <ENTER> to go to the next one]

PSCREEN:

1 only one line of output per time-step (which gives the print out in the previous example)

2 prints out more info.

3 prints out details of the progress of the iterations at each time-step.

`NSCool` at work: good !

Time step number

Time “`t`” (in yrs)

Time increase “`dt`” (in yrs)

Increase in `dt` from previous time step: $\dfrac{dt}{dt^{\mathrm{old}}}$

[`NSCool` is constantly trying to speed up by increasing `dt` from previous time-step]

Iteration:

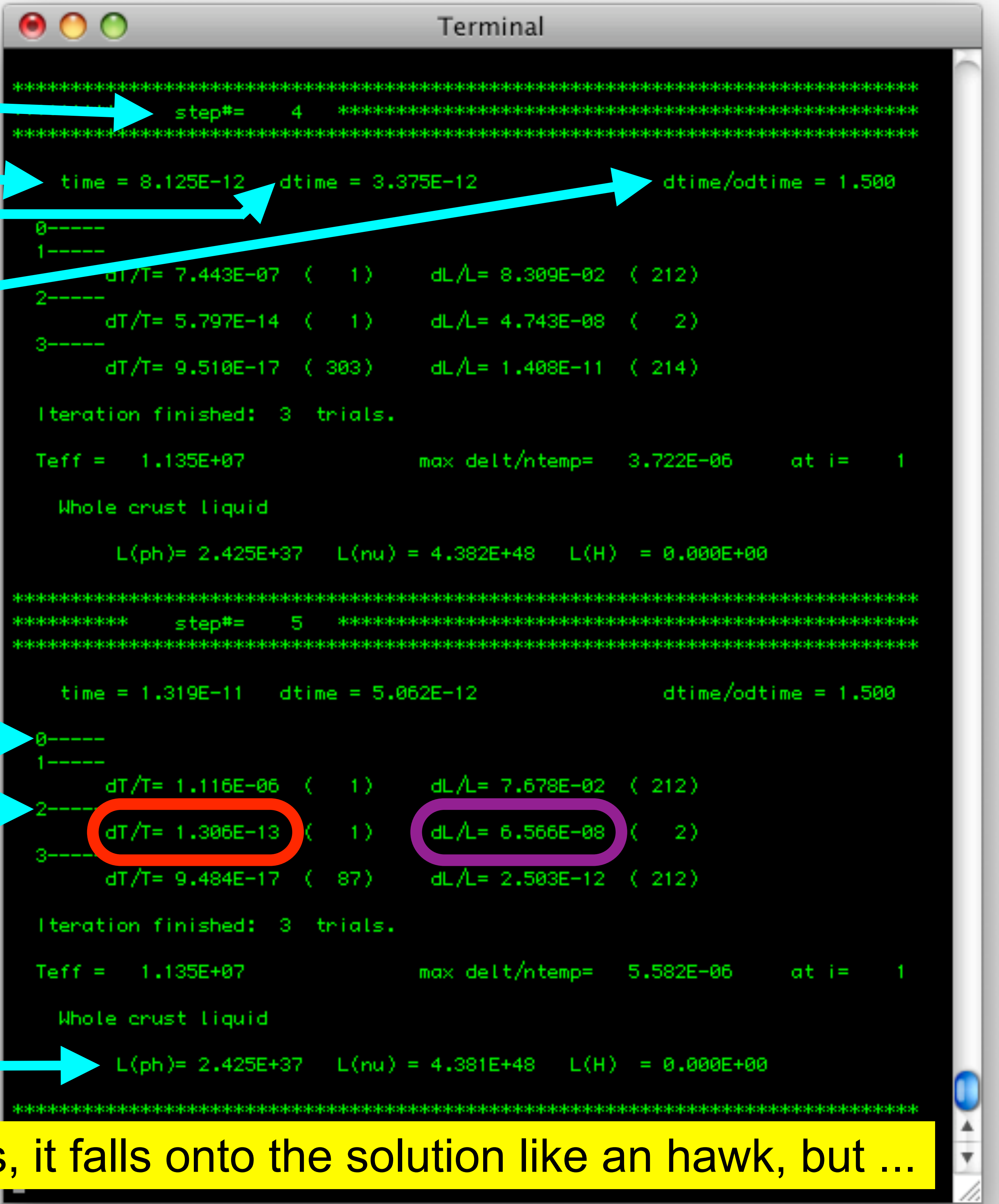
`k = 0` : initial guess (nothing to show)

`k > 0` :

show: $\boxed{\mathrm{Max}_{i=1,3,5,\ldots}\left(\dfrac{\delta T_i^{(k)}}{T_i^{(k)}}\right)}$ $\boxed{\mathrm{Max}_{i=0,2,4,\ldots}\left(\dfrac{\delta \mathcal{L}_i^{(k)}}{\mathcal{L}_i^{(k)}}\right)}$

[In parenthesis is the zone number, “i”, where the Max is reached]

Photon, neutrino and heating luminosities



```
********************************************************************************
**********      step#=     4    ************************************************
********************************************************************************

   time = 8.125E-12    dtime = 3.375E-12                dtime/odtime = 1.500

  0-----
  1-----
        dT/T= 7.443E-07  (    1)     dL/L= 8.309E-02  ( 212)
  2-----
        dT/T= 5.797E-14  (    1)     dL/L= 4.743E-08  (    2)
  3-----
        dT/T= 9.510E-17  ( 303)     dL/L= 1.408E-11  ( 214)

  Iteration finished:  3  trials.

  Teff =   1.135E+07               max delt/ntemp=   3.722E-06     at i=     1

    Whole crust liquid

        L(ph)= 2.425E+37   L(nu) = 4.382E+48   L(H)  = 0.000E+00

********************************************************************************
**********      step#=     5    ************************************************
********************************************************************************

   time = 1.319E-11    dtime = 5.062E-12                dtime/odtime = 1.500

  0-----
  1-----
        dT/T= 1.116E-06  (    1)     dL/L= 7.678E-02  ( 212)
  2-----
        dT/T= 1.306E-13  (    1)     dL/L= 6.566E-08  (    2)
  3-----
        dT/T= 9.484E-17  (   87)     dL/L= 2.503E-12  ( 212)

  Iteration finished:  3  trials.

  Teff =   1.135E+07               max delt/ntemp=   5.582E-06     at i=     1

    Whole crust liquid

        L(ph)= 2.425E+37   L(nu) = 4.381E+48   L(H)  = 0.000E+00

********************************************************************************
```

when it works, it falls onto the solution like an hawk, but ...

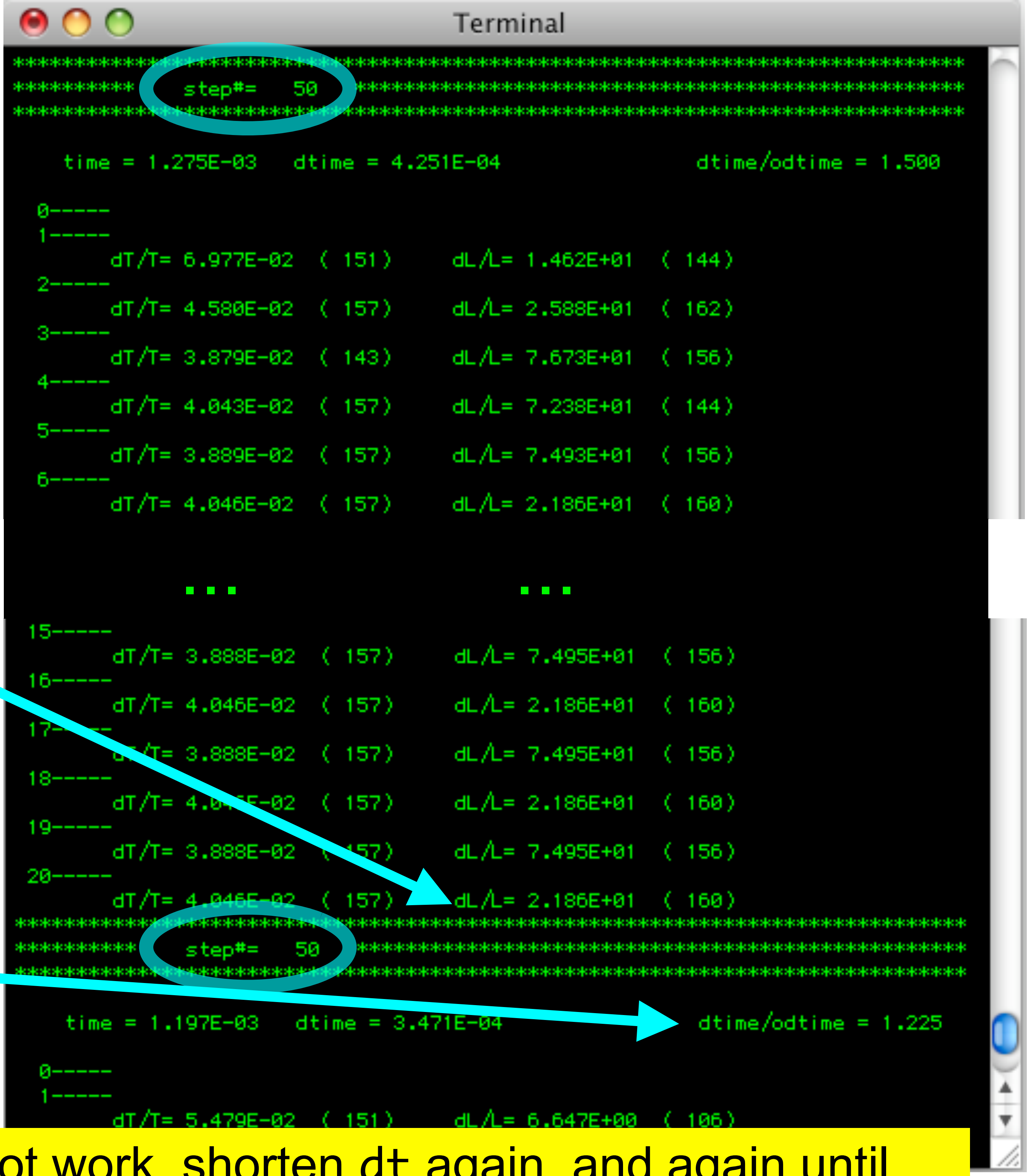
`NSCool` at work: oooops !

When something rough happens inside (as, e.g., the phase transition to superfluidity/ superconductivity) it may become hard to find the new $\mathcal{T}$ and $\mathcal{L}$ profiles:

the initial guess ($\mathcal{T}_i^{(0)}$, $\mathcal{L}_i^{(0)}$) is too far away from the solution ($\mathcal{T}_i$, $\mathcal{L}_i$) and iterations just go nowhere:

$$\mathrm{Max}_{i=0,2,4,\ldots}\left(\frac{\delta\mathcal{L}_i^{(k)}}{\mathcal{L}_i^{(k)}}\right) \simeq 218$$

Remedy: forget these iterations, restart over with a smaller time step `dt`



```
*****************************************************************
**********    step#=    50  **************************************
*****************************************************************

   time = 1.275E-03   dtime = 4.251E-04            dtime/odtime = 1.500

  0-----
  1-----
       dT/T= 6.977E-02  ( 151)    dL/L= 1.462E+01  ( 144)
  2-----
       dT/T= 4.580E-02  ( 157)    dL/L= 2.588E+01  ( 162)
  3-----
       dT/T= 3.879E-02  ( 143)    dL/L= 7.673E+01  ( 156)
  4-----
       dT/T= 4.043E-02  ( 157)    dL/L= 7.238E+01  ( 144)
  5-----
       dT/T= 3.889E-02  ( 157)    dL/L= 7.493E+01  ( 156)
  6-----
       dT/T= 4.046E-02  ( 157)    dL/L= 2.186E+01  ( 160)

             ...                       ...

 15-----
       dT/T= 3.888E-02  ( 157)    dL/L= 7.495E+01  ( 156)
 16-----
       dT/T= 4.046E-02  ( 157)    dL/L= 2.186E+01  ( 160)
 17-----
       dT/T= 3.888E-02  ( 157)    dL/L= 7.495E+01  ( 156)
 18-----
       dT/T= 4.046E-02  ( 157)    dL/L= 2.186E+01  ( 160)
 19-----
       dT/T= 3.888E-02  ( 157)    dL/L= 7.495E+01  ( 156)
 20-----
       dT/T= 4.046E-02  ( 157)    dL/L= 2.186E+01  ( 160)
*****************************************************************
**********    step#=    50  **************************************
*****************************************************************

   time = 1.197E-03   dtime = 3.471E-04            dtime/odtime = 1.225

  0-----
  1-----
       dT/T= 5.479E-02  ( 151)    dL/L= 6.647E+00  ( 106)
```

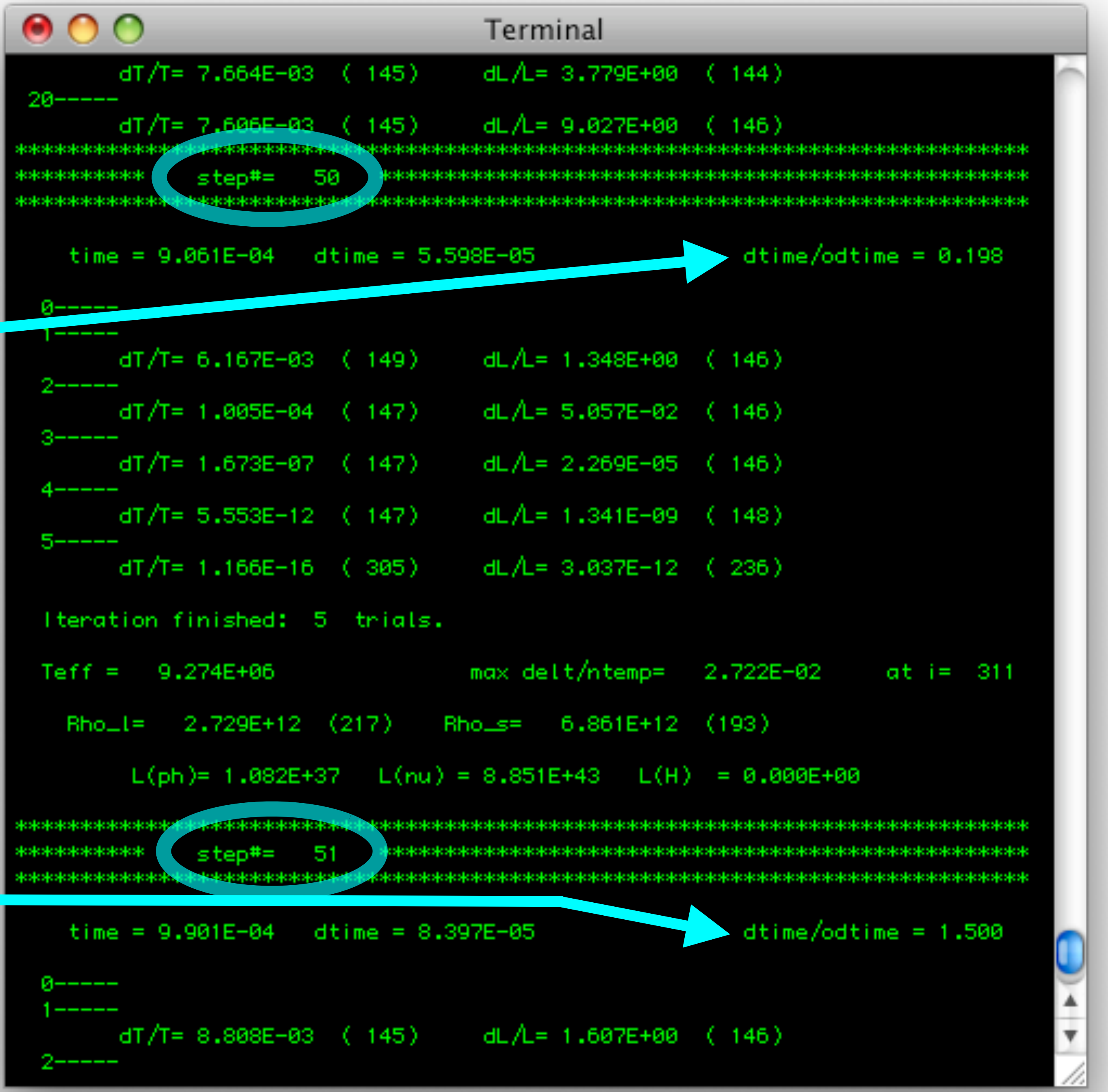
and if it does not work, shorten `dt` again, and again until ...

`NSCool` at work: it works !

Finally, with `dt ~ 0.2 dt`$^{\mathtt{old}}$
the solution is found !

[This ratio is `dt` at step 50 compared to `dt`$^{\mathtt{old}}$ at step 49, not `dt` at the previous iteration attempt of this step 50]

and now, try to rush again ...



```
        dT/T= 7.664E-03  ( 145)     dL/L= 3.779E+00  ( 144)
 20-----
        dT/T= 7.606E-03  ( 145)     dL/L= 9.027E+00  ( 146)
******************************************************************************
**********    step#=   50  ***************************************************
******************************************************************************

    time = 9.061E-04   dtime = 5.598E-05                  dtime/odtime = 0.198

  0-----
  1-----
        dT/T= 6.167E-03  ( 149)     dL/L= 1.348E+00  ( 146)
  2-----
        dT/T= 1.005E-04  ( 147)     dL/L= 5.057E-02  ( 146)
  3-----
        dT/T= 1.673E-07  ( 147)     dL/L= 2.269E-05  ( 146)
  4-----
        dT/T= 5.553E-12  ( 147)     dL/L= 1.341E-09  ( 148)
  5-----
        dT/T= 1.166E-16  ( 305)     dL/L= 3.037E-12  ( 236)

  Iteration finished:  5  trials.

  Teff =   9.274E+06                 max delt/ntemp=   2.722E-02     at i=  311

   Rho_l=   2.729E+12  (217)    Rho_s=   6.861E+12  (193)

        L(ph)= 1.082E+37   L(nu) = 8.851E+43   L(H)  = 0.000E+00

******************************************************************************
**********    step#=   51  ***************************************************
******************************************************************************

    time = 9.901E-04   dtime = 8.397E-05                  dtime/odtime = 1.500

  0-----
  1-----
        dT/T= 8.808E-03  ( 145)     dL/L= 1.607E+00  ( 146)
  2-----
```

It may happen that dt is cut so much that it becomes ~ 0: the only remedy is then

Ctrl-C

and figure out what's happening (very likely you added some weird physics, or a bug)

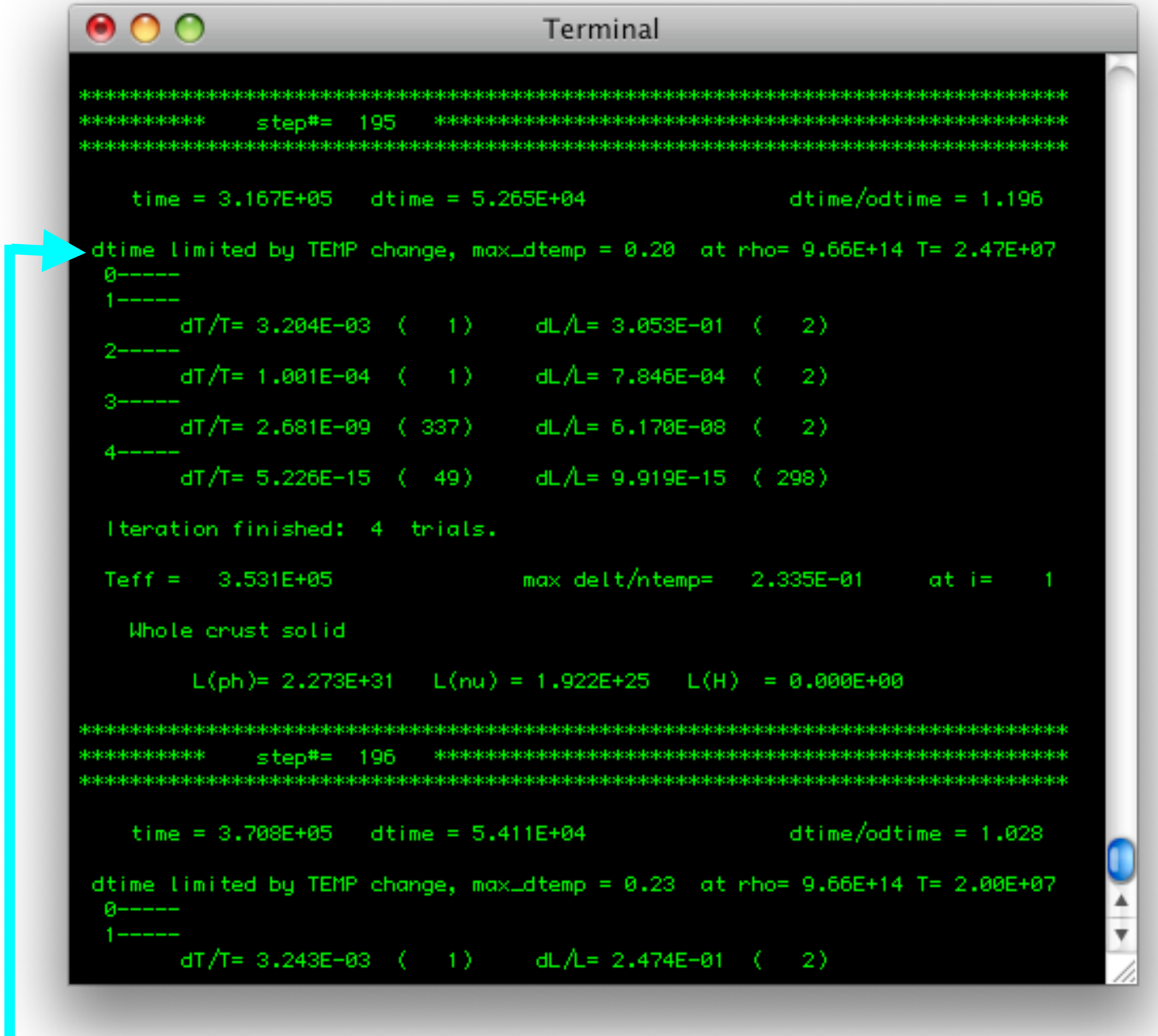
NSCool at work: going too fast !

In this model (an isolated cooling neutron star) when the star gets into photon cooling T decreases much faster that during the neutrino cooling era. Moreover there is not much to calculate because the star is almost perfectly isothermal.
→ NSCool rushes too much !

To avoid loosing accuracy, another test for the time increment dt is that $\mathcal{T}$ does not change too from $\mathcal{T}^{\mathrm{old}}$: if

$$\text{max\_dtemp} \equiv \mathrm{Max}_{i=1,3,5,\ldots} \left(\frac{|\mathcal{T}_i - \mathcal{T}_i^{\mathrm{old}}|}{\mathcal{T}_i^{\mathrm{old}}} \right)$$

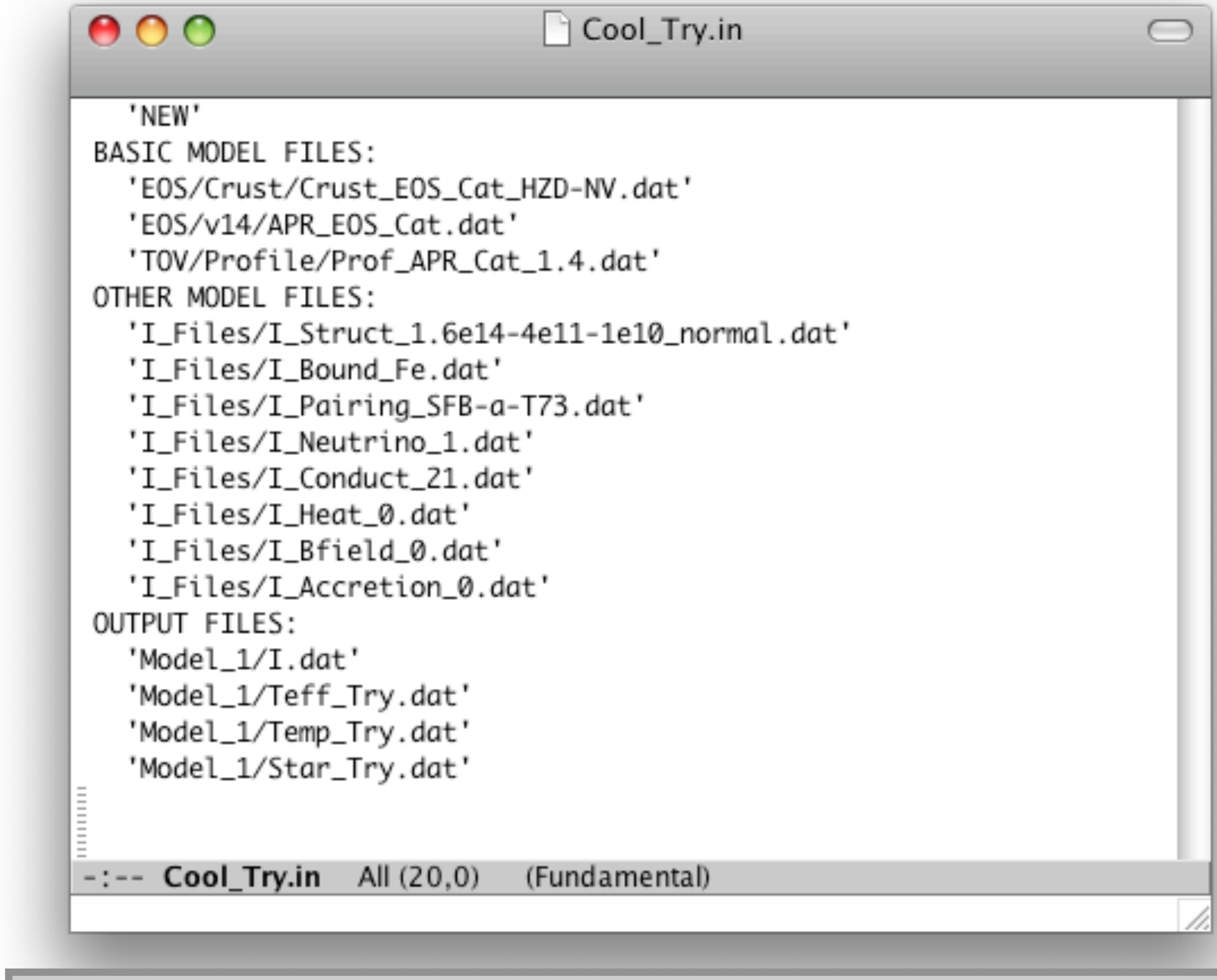
is ≥ 0.2 the increase of dt (with respect to dt^old) is reduced.



```
********************************************************************************
**********     step#=  195    **************************************************
********************************************************************************

   time = 3.167E+05   dtime = 5.265E+04                dtime/odtime = 1.196

 dtime limited by TEMP change, max_dtemp = 0.20  at rho= 9.66E+14 T= 2.47E+07
  0-----
  1-----
        dT/T= 3.204E-03  (    1)     dL/L= 3.053E-01  (    2)
  2-----
        dT/T= 1.001E-04  (    1)     dL/L= 7.846E-04  (    2)
  3-----
        dT/T= 2.681E-09  ( 337)     dL/L= 6.170E-08  (    2)
  4-----
        dT/T= 5.226E-15  (   49)     dL/L= 9.919E-15  ( 298)

  Iteration finished:  4  trials.

  Teff =   3.531E+05                max delt/ntemp=   2.335E-01     at i=    1

    Whole crust solid

       L(ph)= 2.273E+31   L(nu) = 1.922E+25   L(H)  = 0.000E+00

********************************************************************************
**********     step#=  196    **************************************************
********************************************************************************

   time = 3.708E+05   dtime = 5.411E+04                dtime/odtime = 1.028

 dtime limited by TEMP change, max_dtemp = 0.23  at rho= 9.66E+14 T= 2.00E+07
  0-----
  1-----
        dT/T= 3.243E-03  (    1)     dL/L= 2.474E-01  (    2)
```

The Input Master File “`Cool_*.in`”

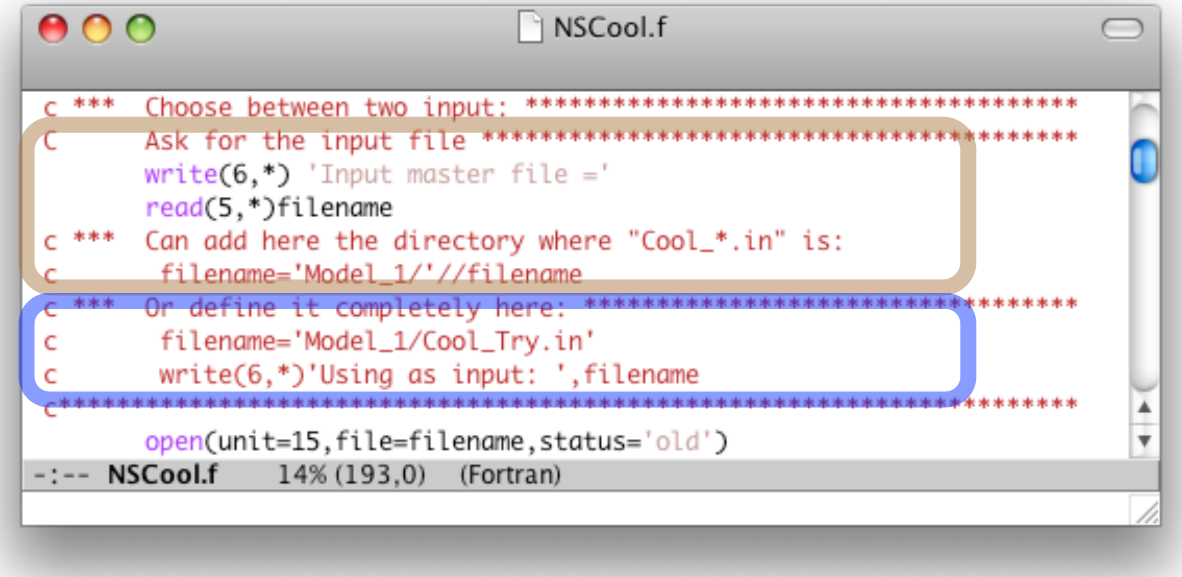
The master input file:



```
   'NEW'
 BASIC MODEL FILES:
   'EOS/Crust/Crust_EOS_Cat_HZD-NV.dat'
   'EOS/v14/APR_EOS_Cat.dat'
   'TOV/Profile/Prof_APR_Cat_1.4.dat'
 OTHER MODEL FILES:
   'I_Files/I_Struct_1.6e14-4e11-1e10_normal.dat'
   'I_Files/I_Bound_Fe.dat'
   'I_Files/I_Pairing_SFB-a-T73.dat'
   'I_Files/I_Neutrino_1.dat'
   'I_Files/I_Conduct_21.dat'
   'I_Files/I_Heat_0.dat'
   'I_Files/I_Bfield_0.dat'
   'I_Files/I_Accretion_0.dat'
 OUTPUT FILES:
   'Model_1/I.dat'
   'Model_1/Teff_Try.dat'
   'Model_1/Temp_Try.dat'
   'Model_1/Star_Try.dat'
```

Its content is briefly described in the next slides.
For more details see: `NSCool_Guide_Control`.

It is defined in `NSCool.f` with:



```
c ***   Choose between two input: ************************************
C       Ask for the input file ****************************************
        write(6,*) 'Input master file ='
        read(5,*)filename
c ***   Can add here the directory where "Cool_*.in" is:
c        filename='Model_1/'//filename
c ***   Or define it completely here: *********************************
c        filename='Model_1/Cool_Try.in'
c        write(6,*)'Using as input: ',filename
c*********************************************************************
        open(unit=15,file=filename,status='old')
```

If you’re tired of typing in the directory, uncomment the
`c    filename=‘Model_1’//filename`

Or, if you’re always using the same input file (e.g., while debugging) use the second version (uncomment it, and comment out the first version)

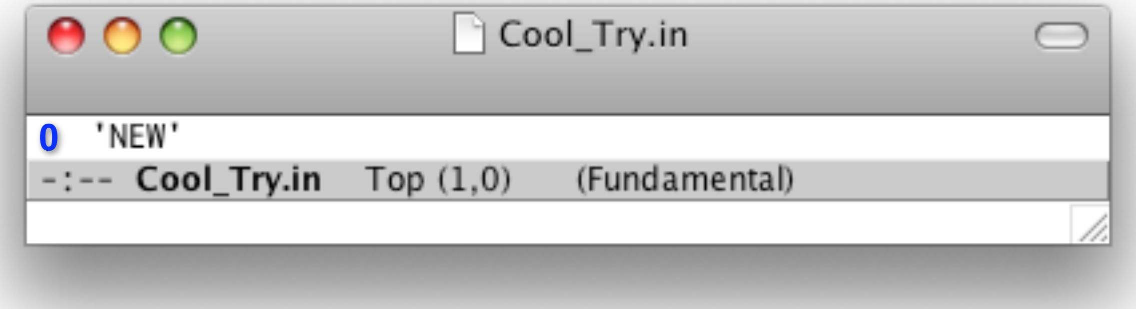
The way it is presently working, NSCool will look for these files as they are defined in the input file.
The way the files are defined in `Cool_Try.in`, since the subdirectories `EOS, TOV, I_Files, Model_1`, ..., are in the `NSCool` directory, one has to run `NSCool` from this directory (and not from the `Code` subdirectory).
But if you define the files as, e.g., ‘**../**`EOS/Crust/Crust_EOS_Cat_HZD-NV.dat`’, you can run from `Code`.

The input file line by line (1)

0: This specifies how much info there is in the core EOS file. Its possible values are:
'old' , 'new' , 'NEW' , 'QRK' , and 'STR'.

Presently, only 'NEW' and 'STR' work properly !

- 'NEW' includes leptons, nucleons and hyperons.
- 'STR' is for strange stars with a thin baryonic crust.
- 'QRK' was as 'new' but with quark matter ("hybrid star"), allowing for a mixed phase. I think it is not anymore compatible with the present version of NSCool, but this should be easy to fix (by making an extension of 'NEW' instead of 'new').



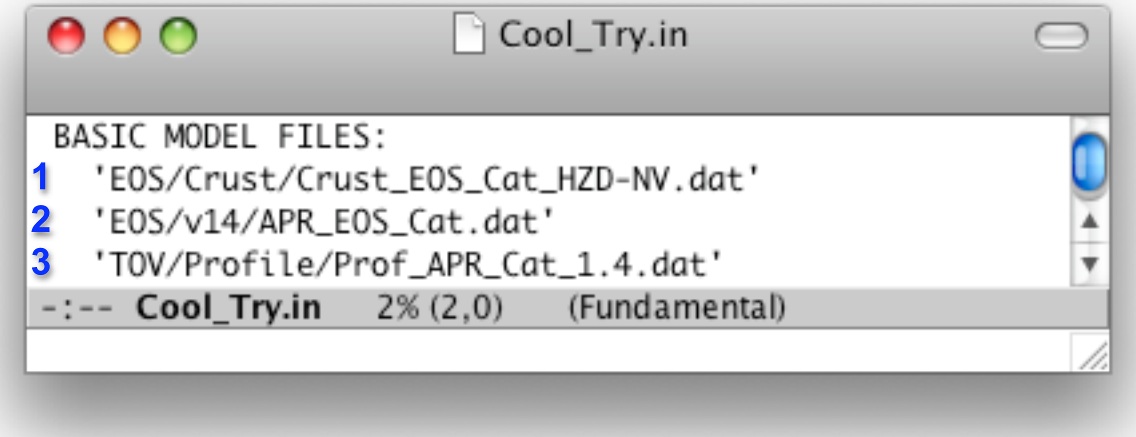
Cool_Try.in

```
0  'NEW'
-:--  Cool_Try.in    Top (1,0)     (Fundamental)
```

- 'old' and 'new' may work, but this needs to be checked.

This terminology obviously reflects the evolution of NSCool with time !

3: contains the profile of the star as calculated by the TOV integrator. Since TOV is integrated by a Runge-Kutta scheme, during initialization a new grid is defined, by interpolations, more appropriate for NSCool.

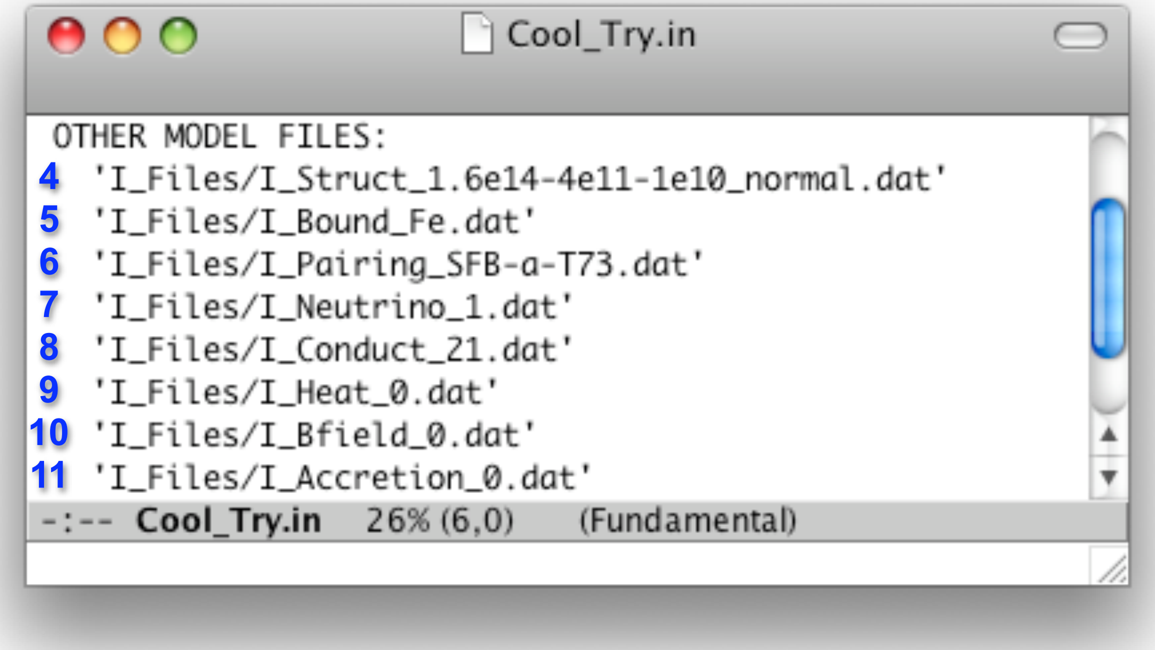


Cool_Try.in

```
 BASIC MODEL FILES:
1  'EOS/Crust/Crust_EOS_Cat_HZD-NV.dat'
2  'EOS/v14/APR_EOS_Cat.dat'
3  'TOV/Profile/Prof_APR_Cat_1.4.dat'
-:--  Cool_Try.in    2% (2,0)     (Fundamental)
```

1: the crust EOS and
2: the core EOS:they are used to define the "chemical composition" of the star, i.e., abundances of each type of particle, and nuclei in the crust, as well as Fermi momenta, effective masses, etc ... All this is performed during initialization.

The input file line by line (2)



```
 OTHER MODEL FILES:
4  'I_Files/I_Struct_1.6e14-4e11-1e10_normal.dat'
5  'I_Files/I_Bound_Fe.dat'
6  'I_Files/I_Pairing_SFB-a-T73.dat'
7  'I_Files/I_Neutrino_1.dat'
8  'I_Files/I_Conduct_21.dat'
9  'I_Files/I_Heat_0.dat'
10 'I_Files/I_Bfield_0.dat'
11 'I_Files/I_Accretion_0.dat'
-:--  Cool_Try.in    26% (6,0)     (Fundamental)
```

4: contains parameters which define the grid used.

I like to give files a name which is pretty indicative of its content. This name indicates that this specific file defines:

$\rho_{\text{crust-core}}$ = 1.6·10^{14} gm cm^{-3}, ρ_{drip} = 4·10^{11} gm cm^{-3}
and ρ_b = 10^{10} gm cm^{-3} (that is the "surface", i.e., the position of the outer boundary), and a "normal" size grid (about 100 grid points in the core and 250 in the crust).

5: defines the outer boundary condition, i.e., the envelope model glued at r_b.

(This particular file defines a heavy element envelope with iron at the surface.)

6: define which pairing gaps are used (for superfluidity/superconductivity).

7: allows some control to turn on or off some neutrino processes.

8: same thing, but for the thermal conductivity.

9: controls which heating mechanisms are used.

10: controls magnetic field evolution: not implemented anymore !

11: controls the accretion rate.

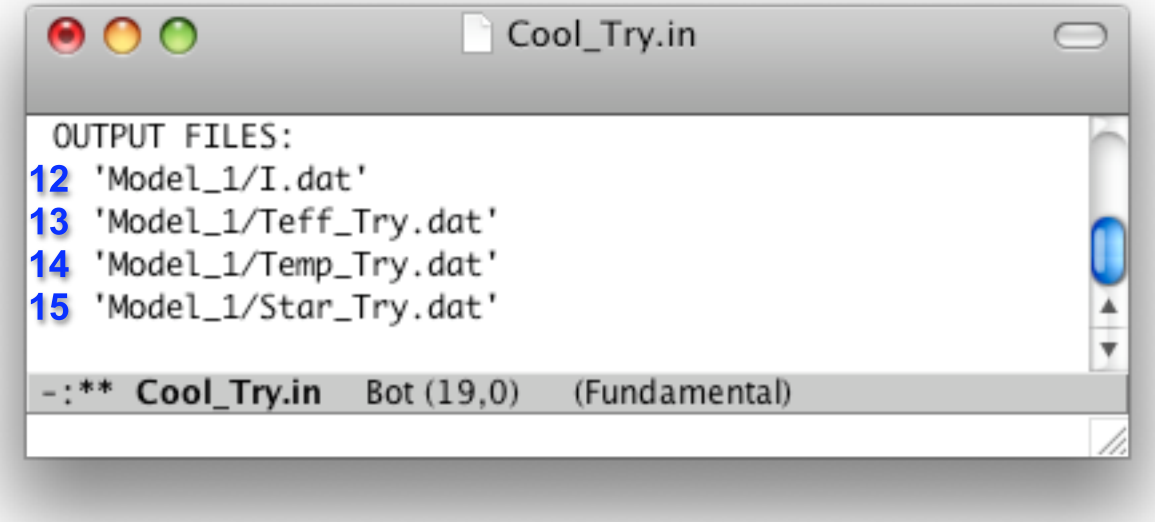
Notice that these last three files have names as I_*_**0**.dat:

this indicates that these files define

- ***no*** heating,
- ***no*** magnetic field evolution (in this case there is no other option !) and
- ***no*** accretion.

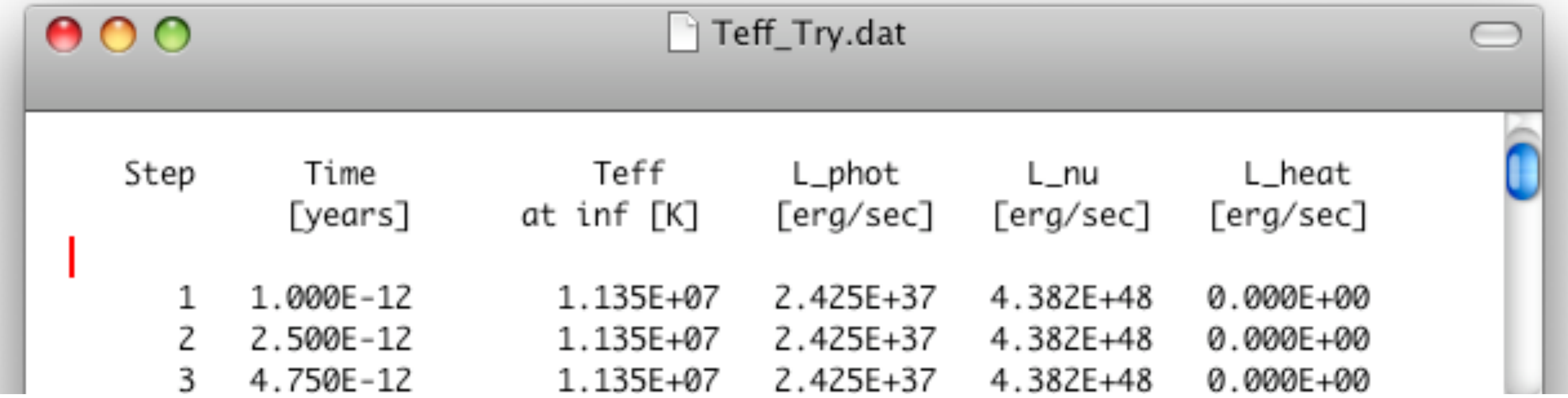
The input file line by line (3)

12: contains some parameter which control the output: see next slide.
[*In this file are the parameters, among others, which determine whether the next three files are used or not.*]



```
 OUTPUT FILES:
12 'Model_1/I.dat'
13 'Model_1/Teff_Try.dat'
14 'Model_1/Temp_Try.dat'
15 'Model_1/Star_Try.dat'

-:** Cool_Try.in     Bot (19,0)    (Fundamental)
```

13: output file: contains a few line of general info about the model and then one line per time step:



```
   Step      Time          Teff      L_phot     L_nu       L_heat
            [years]      at inf [K] [erg/sec]  [erg/sec]  [erg/sec]

      1   1.000E-12     1.135E+07  2.425E+37  4.382E+48  0.000E+00
      2   2.500E-12     1.135E+07  2.425E+37  4.382E+48  0.000E+00
      3   4.750E-12     1.135E+07  2.425E+37  4.382E+48  0.000E+00
      •     •••            •••        •••        •••        •••      etc
```

This is the file to use to plot a cooling curve

14: output file: contains the full *T* & *L* (and other variables) profiles at determined times [the times at which print out is done are given in the file “`I.dat`” **12**]

15: output file: contains the full profile of the star of time independent variables as r, ρ, P, T_C’s, ...

Output control: the "I.dat" file

1: already described previously.

2: DEBUG > 0 will print out zillions of screenful of info about what NSCool is doing. Some hints about it are at the top of the file NSCool.f

3: time step at which the debug print out will begins.

4: PTEFF =1: the output file "Teff_*.dat" is used.

5: PTEMP =1: the output file "Temp_*.dat" is used.

6: PSTAR =1: the output file "Star_*.dat" is used.

7, 8, and 9: some integers which can be used to print out more things in "Teff_*.dat".

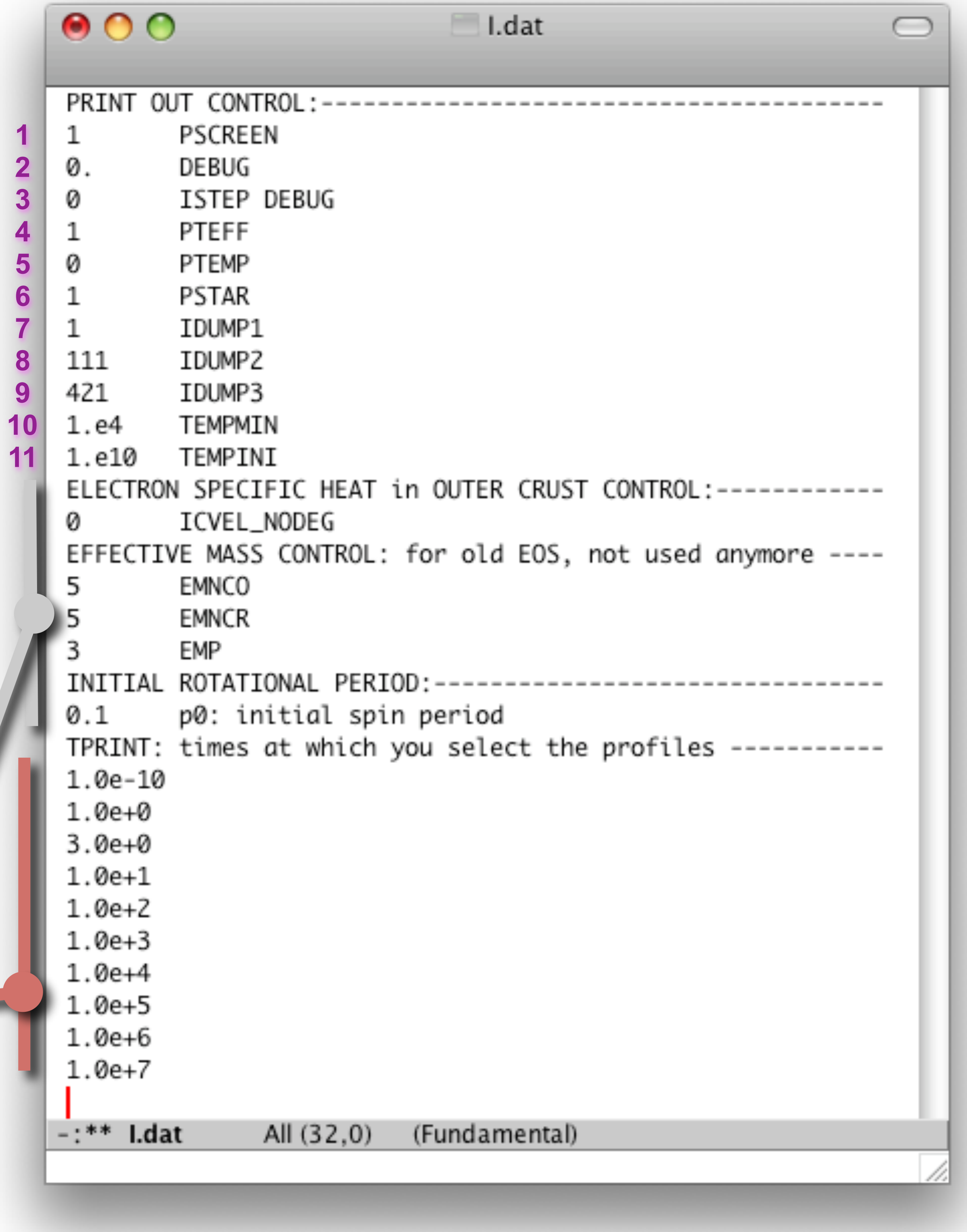
10: the calculation stops when T_e^∞ drops below this value (in K).

11: initial temperature (in K) for the initial T profile.

Not used anymore

[but kept here for backward compatibility]

List of times, in years, at which the full profiles will be printed out in the file "Temp_*.dat".



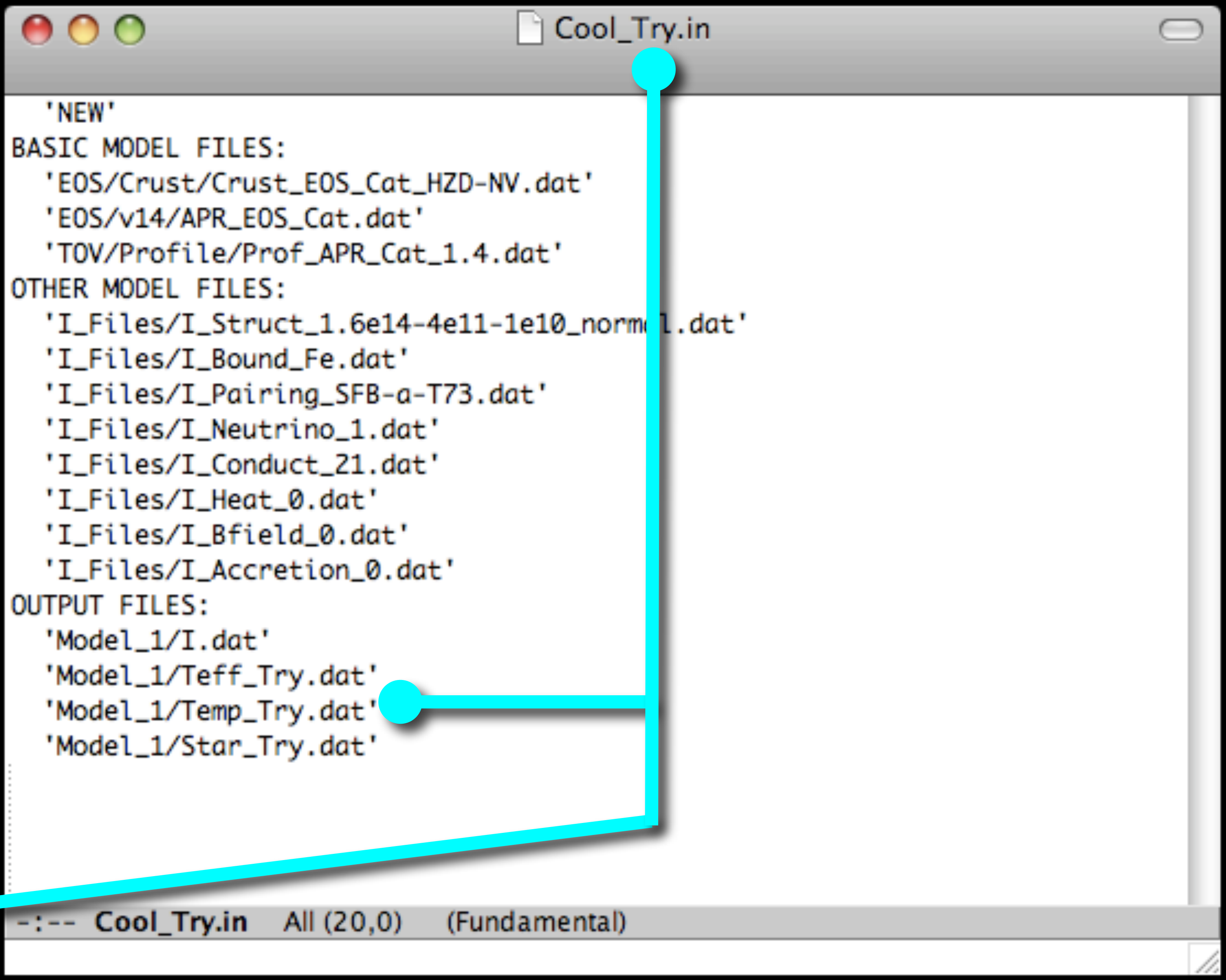
```
   PRINT OUT CONTROL:-------------------------------------------
1  1        PSCREEN
2  0.       DEBUG
3  0        ISTEP DEBUG
4  1        PTEFF
5  0        PTEMP
6  1        PSTAR
7  1        IDUMP1
8  111      IDUMP2
9  421      IDUMP3
10 1.e4     TEMPMIN
11 1.e10    TEMPINI
   ELECTRON SPECIFIC HEAT in OUTER CRUST CONTROL:------------
   0        ICVEL_NODEG
   EFFECTIVE MASS CONTROL: for old EOS, not used anymore ----
   5        EMNCO
   5        EMNCR
   3        EMP
   INITIAL ROTATIONAL PERIOD:-------------------------------
   0.1      p0: initial spin period
   TPRINT: times at which you select the profiles -----------
   1.0e-10
   1.0e+0
   3.0e+0
   1.0e+1
   1.0e+2
   1.0e+3
   1.0e+4
   1.0e+5
   1.0e+6
   1.0e+7

   -:** I.dat        All (32,0)    (Fundamental)
```

Bookkeeping advice

You will rapidly have many many cooling models. It can become a mess.

For easier bookkeeping:

- Create different directories for different families of models, as, e.g., `Model_1, Model_2, Model_3,` ..
- Change the name of the master file `Cool_*.in` for ***each*** model (i.e., each run of `NSCool`) and keep all these files in the subdirectory. They contain all the info about the details of the model !
- Use the same name for the output files `Teff_*.dat, Temp_*.dat` and `Star_*.dat` as for the master `Cool_*.in. Example:`



```
  'NEW'
BASIC MODEL FILES:
  'EOS/Crust/Crust_EOS_Cat_HZD-NV.dat'
  'EOS/v14/APR_EOS_Cat.dat'
  'TOV/Profile/Prof_APR_Cat_1.4.dat'
OTHER MODEL FILES:
  'I_Files/I_Struct_1.6e14-4e11-1e10_normal.dat'
  'I_Files/I_Bound_Fe.dat'
  'I_Files/I_Pairing_SFB-a-T73.dat'
  'I_Files/I_Neutrino_1.dat'
  'I_Files/I_Conduct_21.dat'
  'I_Files/I_Heat_0.dat'
  'I_Files/I_Bfield_0.dat'
  'I_Files/I_Accretion_0.dat'
OUTPUT FILES:
  'Model_1/I.dat'
  'Model_1/Teff_Try.dat'
  'Model_1/Temp_Try.dat'
  'Model_1/Star_Try.dat'
```

DISCLAIMER

Obviously you use this code at your own risks.

WARNING

Nevertheless, most of it is pretty good, but:

Some parts are lousy:

- **Hyperons**: check the neutrino emission subroutines before using them. Moreover there is no hyperon contribution to thermal conductivity.
- **Quarks**: most of the physics is very simplified. Take it as your job to reimplements meaningfull physics according to your needs. There are many subroutines with quark physics, but they have been written at different times, and for different purposes, and it's garanteed that they are mutually inconsistent.

"That's all Folks!"