# Simulating Stars — White Dwarfs and Neutrinos

Ilaria Caiazzo, Jeremy Heyl
TAs: Xianfei Zhang, Sarafina Nance, Ilka Petermann

## 1 Introduction

The cooling of young white dwarfs is dominated by neutrinos, so young white dwarfs are a great probe of weak interactions. In this lab, you are going to extend MESA to vary the neutrino emission rates using `run_star_extras` and a parameter that you can set in the `inlist`.

## 2 `run_star_extras`

**Task:**

1. Make a copy of your working directory from the previous lab to build your neutrino code.
2. You are going to replace the built-in neutrino code with your own code, so go to the directory `$MESA_DIR/star/other`. You are interested in the file called `other_neu.f90`. Copy the subroutine `null_other_neu` from that file into your `run_star_extras.f`. You can copy it anywhere after the word `contains`, and it has to be at the same level as the other subroutines.
3. Compile MESA now by typing `./mk` in your working directory.
4. Rename the subroutine to something else, for example `my_other_neu`. You do this at the top and the bottom of the subroutine.
5. We would like to use a parameter that we can set in the inlist (e.g. `s% x_ctrl(1)`), so we will need to define a variable `s` to hold the `star_ptr` and then copy the `star_ptr` into it.
6. The neutrino rates and their derivatives are held in the array `loss`. You can multiply the entire array by a constant as `loss = s% x_ctrl(1) * loss`. This increases everything by that factor. If your new neutrino rates were more complicated than a simple multiplication, you would have to calculate the new derivatives of loss rates.
7. The array `sources` divides the loss rates by the process that generates them. Let's multiply all of the sources by the same constant. Again, if your additional neutrinos came from a particular process, you could supply that information here.
8. We have to let MESA know which routine to use for the new neutrino rates. The key subroutine in `run_star_extras.f` is called `extras_controls`. It is the first subroutine in the file. We add the command

   ```
   s% other_neu => my_other_neu
   ```

   at the end of the first subroutine which tells MESA to use our routine if we ask for it.
9. Finally we have to tell MESA in the inlist that we want to use a new neutrino routine and the value of our control variable, so we insert the following into the controls section of the inlist

   ```
   use_other_neu = .true.
   x_ctrl(1) = 10
   ```

   to give ten times the neutrino cooling.

**Solution:**

```
subroutine extras_controls(id, ierr)
   integer, intent(in) :: id
   integer, intent(out) :: ierr
   type (star_info), pointer :: s
   ierr = 0
   call star_ptr(id, s, ierr)
   if (ierr /= 0) return

   s% extras_startup => extras_startup
   s% extras_check_model => extras_check_model
   s% extras_finish_step => extras_finish_step
   s% extras_after_evolve => extras_after_evolve
   s% how_many_extra_history_columns => how_many_extra_history_columns
   s% data_for_extra_history_columns => data_for_extra_history_columns
   s% how_many_extra_profile_columns => how_many_extra_profile_columns
   s% data_for_extra_profile_columns => data_for_extra_profile_columns

   s% other_neu => my_other_neu
end subroutine extras_controls
```

```
subroutine my_other_neu(  &
      id, k, T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
      loss, sources, ierr)
   use neu_lib, only: neu_get
   use neu_def
   integer, intent(in) :: id ! id for star
   integer, intent(in) :: k ! cell number or 0 if not for a particular cell
   real(dp), intent(in) :: T ! temperature
   real(dp), intent(in) :: log10_T ! log10 of temperature
   real(dp), intent(in) :: Rho ! density
   real(dp), intent(in) :: log10_Rho ! log10 of density
   real(dp), intent(in) :: abar ! mean atomic weight
   real(dp), intent(in) :: zbar ! mean charge
   real(dp), intent(in) :: z2bar ! mean charge squared
   real(dp), intent(in) :: log10_Tlim
   logical, intent(inout) :: flags(num_neu_types) ! true if should include the type of los
   real(dp), intent(inout) :: loss(num_neu_rvs) ! total from all sources
   real(dp), intent(inout) :: sources(num_neu_types, num_neu_rvs)
   integer, intent(out) :: ierr
   type (star_info), pointer :: s

   ierr=0
   call star_ptr(id, s, ierr)

   call neu_get(  &
      T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
      loss, sources, ierr)
   loss=s%x_ctrl(1)*loss
   sources=s%x_ctrl(1)*sources
end subroutine null_other_neu
```

# 3 Evolution

**Task:**

1. Run the evolution of the best-fitting model from the previous lab with various neutrino rates, 0.1, 0.3, 3 and 10 times the standard rate.
2. Use `paintisochone.py` to calculate the absolute magnitudes of your model white dwarfs.

**Hint:**
You may have to change the number of retries, backups and varcontrol to get the new models to work.

# 4 Analysis

**Task:**

1. Plot luminosity against effective temperature. Does varying the neutrinos have an effect?
2. Plot luminosity against core temperature. Does varying the neutrinos have an effect?
3. Plot luminosity against time. Does varying the neutrinos have an effect?
4. The cumulative luminosity function measures the cooling evolution of the white dwarfs. The young ones are bright in the ultraviolet. Plot the cumulative luminosity functions of the observed white dwarfs in the two fields with your new models.
5. Remember to add the distance modulus, birth-rate estimate and possibly a shift in the age of the white dwarfs as you did in the earlier lab.