MESA

Modules for Experiments in Stellar Astrophysics

Latest News

- 21 Mar 2018
  » Release 10398
- 28 Jan 2018
  » New MESA SDK Version
- 13 Jan 2018
  » Summer School 2018
- 23 Oct 2017
  » Release 10108
- 23 Oct 2017
  » Instrument Paper 4
- 21 Sep 2017
  » New MESA SDK Version
- 12 Sep 2017
  » Release 10000
- 01 Sep 2017
  » Docker, Python and Web
- 14 Aug 2017
  » New mesa-users list
- 02 Aug 2017
  » New MESA SDK Version

This web documentation corresponds to the most recent MESA release (r10398). Documentation for past versions can be found here.

If you find errors or formatting issues, please email Josiah Schwab using this link.

This page documents the MESA options that are part of the controls namelist. It is autogenerated from the file $MESA_DIR/star/defaults/controls.defaults.

Boxes like

```
option_name = 'default'
```

show the default value of each option. To override the default values, add an entry to the controls namelist in your inlist.

Contents

specifications for starting model ¶

NOTE: if you are loading a saved model, then the following initial values are NOT USED to modify the model. in particular, you cannot use these to change Y or Z of an existing model. if you want to do that, see `star_job.defaults` controls such as `change_Y`. however, these are reported in output as the initial values for the star.

initial_mass ¶

initial mass in Msun units. can be any value you'd like when you are creating a pre-main sequence model.

not used when loading a saved model. however is reported in output as the initial mass of the star.

if you are loading a ZAMS model and the requested mass is in the range of prebuilt models, the code will interpolate in mass using the closest prebuilt models. if the requested mass is beyond the range of the prebuilt models, the code will load the closest one and then call "relax mass" to create a model to match the request. the prebuilt range is 0.08 Msun to 100 Msun, so the `relax_mass` method is only used for extreme cases. there are enough prebuilt models that the interpolation in mass seems to work fine for many applications.

```
initial_mass = 1
```

initial_z ¶

initial metallicity for create pre-ms and create initial model `initial_z` can be any value from 0 to 0.04

not used when loading a saved model. however is reported in output as the initial Z of the star.

however, if you are loading a zams model, then `initial_z` must match one of the prebuilt values. look in the `'data/star_data/zams_models'` directory to see what prebuilt zams Z's are available. at time of writing, only 0.02 was included in the standard version of star.

```
initial_z = 0.02d0
```

initial_y ¶

initial helium mass fraction for create pre-ms and create initial (< 0 means use default which is 0.24 + 2*initial_z)

not used when loading a saved model or a zams model. however is reported in output as the initial Y of the star.

NOTE: this is only used for create pre-main-sequence model and create initial model, and not when loading a zams model.

```
initial_y = -1
```

controls for output ¶

terminal_interval ¶

write info to terminal when `mod(model_number, terminal_interval) = 0`. note: this replaces the obsolete control `terminal_cnt`.

```
terminal_interval = 1
```

write_header_frequency ¶

output the log header info to the terminal when `mod(model_number, write_header_frequency*terminal_interval) = 0`.

```
write_header_frequency = 10
```

extra_terminal_output_file ¶

if not empty, output terminal info to this file in addition to terminal. this does not capture all of the terminal output – just the common items. it is intended for use in situations where you cannot directly see the terminal output such as when running on a cluster. if you want to be able to monitor the progress for such cases, you can set `extra_terminal_output_file = 'log'` and then do `tail -f log` to view the terminal output as it is recorded in the file.

```
extra_terminal_output_file = ''
```

terminal_show_age_in_years ¶

if false, then show in seconds

```
terminal_show_age_in_years = .true.
```

terminal_show_age_in_days ¶

```
terminal_show_age_in_days = .false.
```

num_trace_history_values ¶

any valid name for a history data column, such as `surf_v_rot` for example if you have rapid rotation at the surface, you might want to try something like this:

```
num_trace_history_values = 7
trace_history_value_name(1) = 'surf_v_rot'
trace_history_value_name(2) = 'surf_omega_div_omega_crit'
```

```
trace_history_value_name(3) = 'log_rotational_mdot_boost'
trace_history_value_name(4) = 'log_total_angular_momentum'
trace_history_value_name(5) = 'center n14'
trace_history_value_name(6) = 'surface n14'
trace_history_value_name(7) = 'average n14'
```

value must be less than or equal to 10

```
num_trace_history_values = 0
```

trace_history_value_name(:) ¶

write values to terminal

```
trace_history_value_name(:) = ''
```

photo_directory ¶

directory for binary snapshots used in restarts

```
photo_directory = 'photos'
```

photo_interval ¶

save a photo file for possible restarting when `mod(model_number, photo_interval) = 0`. note: this replaces the obsolete control `photostep`.

```
photo_interval = 50
```

photo_digits ¶

use this many digits from the end of the `model_number` for the photo name

```
photo_digits = 3
```

log_directory ¶

for data files about the run

```
log_directory = 'LOGS'
```

do_history_file ¶

history file is created if this is true

```
do_history_file = .true.
```

history_interval ¶

append an entry to the history.data file when `mod(model_number, history_interval) = 0`.

```
history_interval = 5
```

star_history_name ¶

name of history file

```
star_history_name = 'history.data'
```

star_history_header_name ¶

If not empty, then put star history header info in `star_history_name` file. In this case the history file has only data, making it easier to use with some plotting packages.

```
star_history_header_name = ''
```

star_history_dbl_format ¶

format for writing reals to `star_history_name` file

```
star_history_dbl_format = '(1pes40.16e3, 1x)'
```

star_history_int_format ¶

format for writing integer to `star_history_name` file

```
star_history_int_format = '(i40, 1x)'
```

star_history_txt_format ¶

format for writing characters to `star_history_name` file

```
star_history_txt_format = '(a40, 1x)'
```

write_profiles_flag ¶

profiles are written only if this is true

```
write_profiles_flag = .true.
```

profile_interval ¶

save a model profile info when `mod(model_number, profile_interval) = 0`.

```
profile_interval = 50
```

priority_profile_interval ¶

give saved profile a higher priority for retention when `mod(model_number, priority_profile_interval) = 0`.

```
priority_profile_interval = 1000
```

profiles_index_name ¶

name of the profile index file

```
profiles_index_name = 'profiles.index'
```

profiles_data_prefix ¶

prefix of the profile data

```
profile_data_prefix = 'profile'
```

profiles_data_suffix ¶

suffix of the profile data

```
profile_data_suffix = '.data'
```

profile_data_header_suffix ¶

If not empty, then put profile data header info here. In this case the profile data file has only data, making it easier to use with some plotting packages.

```
profile_data_header_suffix = ''
```

profile_dbl_format ¶

format for writing reals to profile file

```
profile_dbl_format = '(1pes40.16e3, 1x)'
```

profile_int_format ¶

format for writing integers to profile file

```
profile_int_format = '(i40, 1x)'
```

profile_txt_format ¶

format for writing characters to profile file

```
profile_txt_format = '(a40, 1x)'
```

max_num_profile_zones ¶

if `nz > this`, then only write a subsample of the zones. only used if > 1

```
max_num_profile_zones = -1
```

max_num_profile_models ¶

Maximum number of saved profiles. If there's no limit on the number of profiles saved, you can fill up your disk – I've done it. So it's a good idea to set this limit to a reasonable number such as 20 or 30. Once that many have been saved during a run, old ones will be discarded to make room for new ones. Profiles that were saved for key events are given priority and aren't removed as long as there is a lower priority profile that can be discarded instead. Less than zero means no limit.

```
max_num_profile_models = 100
```

profile_model ¶

save profile when `model_number` equals this

```
profile_model = -1
```

write_model_with_profile ¶

if this is true, models are written at same time as profiles

```
write_model_with_profile = .false.
```

model_data_prefix ¶

prefix of the model data files

```
model_data_prefix = 'profile'
```

model_data_suffix ¶

suffix of the model data files

```
model_data_suffix = '.mod'
```

write_controls_info_with_profile ¶

if this is true, the values of the options in the controls inlist are written at same time as profiles

```
write_controls_info_with_profile = .false.
```

controls_data_prefix ¶

prefix of the control data files

```
controls_data_prefix = 'controls'
```

controls_data_suffix ¶

suffix of the control data files

```
controls_data_suffix = '.data'
```

mixing_D_limit_for_log ¶

if max `D_mix` in mixing region is less than this, don't include the region in the log doesn't apply to thermohaline or semiconvective regions

```
mixing_D_limit_for_log = 1d4
```

eta_RTI_limit_for_log ¶

if `eta_RTI` is less than this, treat it as zero for history log

```
eta_RTI_limit_for_log = 1d4
```

mass_loc_for_extra_log_info ¶

log contains info about this mass location in the model negative value means “don’t bother”

```
mass_loc_for_extra_log_info = -1
```

write_pulse_data_with_profile ¶

if true, write pulse info file when writing profile

```
write_pulse_data_with_profile = .false.
```

pulse_data_format ¶

pulsation code format, e.g., ‘FGONG’, ‘OSC’, ‘GYRE’

```
pulse_data_format = 'FGONG'
```

add_atmosphere_to_pulse_data ¶

if true, write atmosphere to pulse files

```
add_atmosphere_to_pulse_data = .false.
```

add_center_point_to_pulse_data ¶

if true, add point for r=0 to pulse files

```
add_center_point_to_pulse_data = .true.
```

keep_surface_point_for_pulse_data ¶

if true, add k=1 cell to pulse files

```
keep_surface_point_for_pulse_data = .false.
```

add_double_points_to_pulse_data ¶

add double points at discontinuities

```
add_double_points_to_pulse_data = .false.
```

interpolate_rho_for_pulse_data ¶

If true, then get `rho_face` by interpolating rho at cell center. If false, then calculate rho_face by `dm/(4*pi*r^2*dr)`.

```
interpolate_rho_for_pulse_data = .true.
```

threshold_grad_mu_for_double_point ¶

threshold in grad_mu = dln(mu)/dln(P) for a double point to be written

```
threshold_grad_mu_for_double_point = 10d0
```

max_number_of_double_points ¶

maximum number of double points to be written (0 = no limit); when this limit is set, double points are chosen in order of decreasing |grad_mu|

```
max_number_of_double_points = 0
```

format_for_FGONG_data ¶

This is the 'wide' FGONG format, as agreed on at the 5th Aarhus RGB workshop (University of Birmingham, UK, October 2015)

```
format_for_FGONG_data = '(1P,5(X,E26.18E3))'
```

format_for_OSC_data ¶

[FGONG Format Documentation] (http://www.astro.up.pt/corot/ntools/docs/CoRoT_ESTA_Files.pdf)

```
format_for_OSC_data = '(1P5E19.12,x)'
```

write_pulsation_plot_data ¶

if true and saving pulsation info, also write out text file in column format for plotting

```
write_pulsation_plot_data = .false.
```

max_num_gyre_points ¶

limit gyre output files to at most this number of points only used when > 1

```
max_num_gyre_points = -1
```

fgong_zero_A_inside_r ¶

when writing FGONG, if r < this and cell has mixing of some kind, force A = 0 Rsun units

```
fgong_zero_A_inside_r = 0d0
```

trace_mass_location ¶

location for `trace_mass_radius`, `trace_mass_logT`, etc. (Msun units)

```
trace_mass_location = 0
```

min_tau_for_max_abs_v_location ¶

controls choice of location in model for `max_abs_v` history info. can use this to exclude locations too close to surface. ignore if <= 0

```
min_tau_for_max_abs_v_location = 0
```

min_q_for_inner_mach1_location ¶

controls choice of location in model for innermost mach 1 history info. can use this to exclude locations too close to center.

```
min_q_for_inner_mach1_location = 0
```

max_q_for_outer_mach1_location ¶

controls choice of location in model for outermost mach 1 history info. can use this to exclude locations too close to surface.

```
max_q_for_outer_mach1_location = 1
```

burn_min1 ¶

used for reporting where burning zone occur, for example in the pgstar TRho profiles. see `star/public/star_data.inc` for details. must be < `burn_min2`. In ergs/g/sec.

```
burn_min1 = 50
```

burn_min2 ¶

used for reporting where burning zone occur, for example in the pgstar TRho profiles. see `star/public/star_data.inc` for details. In ergs/g/sec.

```
burn_min2 = 1000
```

max_conv_vel_div_csound_maxq ¶

only consider from center out to this location

```
max_conv_vel_div_csound_maxq = 1
```

width_for_limit_conv_vel ¶

look this number of cells on either side of boundary to see if any boundary k in that range has s% csound(k) < s% v(k) <= s% csound(k-1) i.e. transition from subsonic to supersonic as go inward if find any such transition then don't allow increase in convection velocity. this implies no change from radiative to convective. the purpose of this is to prevent convective energy transport from moving energy from behind a shock to in front of the shock.

```
width_for_limit_conv_vel = 3
```

max_q_for_limit_conv_vel ¶

for q(k) <= this, don't allow conv_vel to grow

```
max_q_for_limit_conv_vel = -1
```

max_r_in_cm_for_limit_conv_vel ¶

for r(k) <= this, don't allow conv_vel to grow

```
max_r_in_cm_for_limit_conv_vel = -1
```

max_mass_in_gm_for_limit_conv_vel ¶

for m(k) <= this, don't allow conv_vel to grow

```
max_mass_in_gm_for_limit_conv_vel = -1
```

center_avg_value_dq ¶

reported center values are averages over this fraction of star mass

```
center_avg_value_dq = 1d-8
```

surface_avg_abundance_dq ¶

reported surface abundances are averages over this fraction of star mass

```
surface_avg_abundance_dq = 1d-8
```

mass_depth_for_L_surf ¶

only if `use_flux_limiting_with_dPrad_dm_form`

```
mass_depth_for_L_surf = 0d0
```

conv_core_gap_dq_limit ¶

skip non-convective gaps of less than this limit when reporting convective core size

```
conv_core_gap_dq_limit = 0d0
```

definition of core overshooting boundary for output ¶

alpha_bdy_core_overshooting ¶

```
bdy = core_overshoot_r0 + core_overshoot_Hp* &
    (alpha_bdy_core_overshooting*core_overshoot_f - core_overshoot_f0)
```

```
alpha_bdy_core_overshooting = 5
```

definition of core boundaries ¶

he_core_boundary_h1_fraction ¶

If >= 0, boundary is outermost location where h1 mass fraction is <= this value, and he4 mass fraction >= `min_boundary_fraction` (see below). If < 0, boundary is outermost location where he4 is the most abundant species.

```
he_core_boundary_h1_fraction = 0.01d0
```

c_core_boundary_he4_fraction ¶

If >= 0, boundary is outermost location where he4 mass fraction is <= this value, and c12 mass fraction >= `min_boundary_fraction` (see below). If < 0, boundary is outermost location where c12 is the most

abundant species.

```
c_core_boundary_he4_fraction = 0.01d0
```

o_core_boundary_c12_fraction ¶

If >= 0, boundary is outermost location where c12 mass fraction is <= this value, and o16 mass fraction >= `min_boundary_fraction` (see below). If < 0, boundary is outermost location where o16 is the most abundant species.

```
o_core_boundary_c12_fraction = 0.01d0
```

si_core_boundary_o16_fraction ¶

If >= 0, boundary is outermost location where o16 mass fraction is <= this value, and si28 mass fraction >= `min_boundary_fraction` (see below). If < 0, boundary is outermost location where si28 is the most abundant species.

```
si_core_boundary_o16_fraction = 0.01d0
```

fe_core_boundary_si28_fraction ¶

For this case, “iron” includes any species with A > 46. If >= 0, boundary is outermost location where si28 mass fraction is <= this value, and “iron” mass fraction >= `min_boundary_fraction` (see below). If < 0, boundary is outermost location where “iron” is the most abundant species.

```
fe_core_boundary_si28_fraction = 0.01d0
```

neutron_rich_core_boundary_Ye_max ¶

Boundary is outermost location where Ye is <= this value.

```
neutron_rich_core_boundary_Ye_max = 0.48d0
```

min_boundary_fraction ¶

Value for deciding boundary regions.

```
min_boundary_fraction = 0.1d0
```

when to stop ¶

max_model_number ¶

The code will stop when it reaches this model number. Negative means no maximum.

```
max_model_number = -1
```

when_to_stop_rtol ¶

Relative error criteria when hitting stop target time. The system will automatically redo with a smaller timestep to hit a stopping target. It calculates the following “error” term and retries if it is > 1.

```
error = abs(value - target_value)/ &
        (when_to_stop_atol + when_to_stop_rtol*max(abs(value),abs(targ
```

```
when_to_stop_rtol = 1d99
```

when_to_stop_atol ¶

Abolute error criteria when hitting stop target time. The system will automatically redo with a smaller timestep to hit a stopping target. It calculates the following “error” term and retries if it is > 1.

```
error = abs(value - target_value)/ &
        (when_to_stop_atol + when_to_stop_rtol*max(abs(value),abs(targ
```

```
when_to_stop_atol = 1d99
```

max_age ¶

Stop when the age of the star exceeds this value (in years). only applies when > 0.

```
max_age = 1d36
```

max_age_in_seconds ¶

Stop when the age of the star exceeds this value (in seconds). only applies when > 0.

```
max_age_in_seconds = -1
```

num_adjusted_dt_steps_before_max_age ¶

This adjusts `max_years_for_timestep` so that hit `max_age` exactly, without needing possibly large change in timestep at end of run. only used if > 0

number of time steps to adjust to prior to hitting max age only used if > 0

```
num_adjusted_dt_steps_before_max_age = 0
```

dt_years_for_steps_before_max_age ¶

timestep in years

```
dt_years_for_steps_before_max_age = 1d6
```

reduction_factor_for_max_timestep ¶

per time step reduction limited to this

```
reduction_factor_for_max_timestep = 0.98d0
```

gamma_center_limit ¶

gamma is the plasma interaction parameter. Stop when the center value of gamma exceeds this limit.

```
gamma_center_limit = 1d99
```

eta_center_limit ¶

eta is the electron chemical potential in units of k*T. Stop when the center value of eta exceeds this limit.

```
eta_center_limit = 1d99
```

log_center_density_limit ¶

Stop when log10 of the center density exceeds this limit.

```
log_center_density_limit = 11
```

log_center_density_lower_limit ¶

Stop when log10 of the center density is below this limit.

```
log_center_density_lower_limit = -1d99
```

log_center_temp_limit ¶

Stop when log10 of the center temperature exceeds this limit.

```
log_center_temp_limit = 11
```

log_center_temp_lower_limit ¶

Stop when log10 of the center temperature is below this limit.

```
log_center_temp_lower_limit = -1d99
```

surface_accel_div_grav_limit = -1 ¶

This is used when do not have a velocity variable. The acceleration ratio is `abs(accel)/grav` at surface, where accel is `(rdot-rdot_old)/dt` and grav is `G*m/r^2`. Stop if the ratio becomes larger than this limit. Ignored if <= 0.

```
surface_accel_div_grav_limit = -1
```

log_max_temp_upper_limit ¶

stop when log10 of the maximum temperature rises above this limit.

```
log_max_temp_upper_limit = 99
```

log_max_temp_lower_limit ¶

stop when log10 of the maximum temperature drops below this limit.

```
log_max_temp_lower_limit = -99
```

center_entropy_limit ¶

stop when the center entropy exceeds this limit. in kerg per baryon

```
center_entropy_limit = 1d99
```

center_entropy_lower_limit ¶

stop when the center entropy is below this limit. in kerg per baryon

```
center_entropy_lower_limit = -1d99
```

max_entropy_limit ¶

stop when the max entropy exceeds this limit. in kerg per baryon

```
max_entropy_limit = 1d99
```

max_entropy_lower_limit ¶

stop when the max entropy is below this limit. in kerg per baryon

```
max_entropy_lower_limit = -1d99
```

xa_central_lower_limit_species ¶

xa_central_lower_limit ¶

Lower limits on central mass fractions. Stop when central abundance drops below this limit. Can have up to `num_xa_central_limits` of these (see `star_def.inc` for value).
`xa_central_lower_limit_species` contains an isotope name as defined in `chem_def.f`.
`xa_central_lower_limit` contains the lower limit value.

```
xa_central_lower_limit_species(1) = ''
xa_central_lower_limit(1) = 0
```

xa_central_upper_limit_species ¶

xa_central_upper_limit ¶

Upper limits on central mass fractions. Stop when central abundance rises above this limit. Can have up to `num_xa_central_limits` of these (see `star_def.inc` for value). E.g., to stop when center c12 abundance reaches 0.5, set

```
xa_central_upper_limit_species(1) = 'c12'
xa_central_upper_limit(1) = 0.5
```

```
xa_central_upper_limit_species(1) = ''
xa_central_upper_limit(1) = 0
```

xa_surface_lower_limit_species ¶

xa_surface_lower_limit ¶

Lower limits on surface mass fractions. Stop when surface abundance drops below this limit. Can have up to `num_xa_surface_limits` of these (see `star_def` for value)
`xa_surface_lower_limit_species` contains an isotope name as defined in `chem_def.f`
`xa_surface_lower_limit` contains the lower limit value

```
xa_surface_lower_limit_species(1) = ''
xa_surface_lower_limit(1) = 0
```

xa_surface_upper_limit_species ¶

xa_surface_upper_limit ¶

upper limits on surface mass fractions stop when surface abundance rises above this limit can have up to `num_xa_surface_limits` of these (see `star_def` for value) e.g., to stop when surface c12 abundance reaches 0.5, set

```
xa_surface_upper_limit_species(1) = 'c12'
xa_surface_upper_limit(1) = 0.5
```

```
xa_surface_upper_limit_species(1) = ''
xa_surface_upper_limit(1) = 0
```

xa_average_lower_limit_species ¶

xa_average_lower_limit ¶

lower limits on average mass fractions stop when average abundance drops below this limit can have up to `num_xa_average_limits` of these (see `star_def` for value)

```
xa_average_lower_limit_species(1) = ''
xa_average_lower_limit(1) = 0
```

xa_average_upper_limit_species ¶

xa_average_upper_limit ¶

upper limits on average mass fractions stop when average abundance rises above this limit can have up to `num_xa_average_limits` of these (see `star_def` for value)

```
xa_average_upper_limit_species(1) = ''
xa_average_upper_limit(1) = 0
```

HB_limit ¶

For detecting horizontal branch. Only applies when center abundance by mass of h1 is < 1d-4. Stop when the center abundance by mass of he4 drops below this limit.

```
HB_limit = 0
```

stop_at_TP ¶

If true, stop at next AGB thermal pulse. This is defined as having a convective zone with helium burning when central helium is depleted and `he_core_mass - c_core_mass <= TP_he_shell_max`.

```
stop_at_TP = .false.
```

TP_he_shell_max ¶

Stop when thermal pulse helium shell mass reaches this value, in Msun units

```
TP_he_shell_max = 0.2d0
```

star_mass_min_limit ¶

Stop when star mass in Msun units is < this. <= 0 means no limit.

```
star_mass_min_limit = 0
```

star_mass_max_limit ¶

Stop when star mass in Msun units is > this. <= 0 means no limit.

```
star_mass_max_limit = 0
```

star_H_mass_min_limit ¶

Stop when star hydrogen mass in Msun units is < this. <= 0 means no limit.

```
star_H_mass_min_limit = 0
```

star_H_mass_max_limit ¶

Stop when star hydrogen mass in Msun units is > this. <= 0 means no limit.

```
star_H_mass_max_limit = 0
```

star_He_mass_min_limit ¶

Stop when star he3+he4 mass in Msun units is < this. <= 0 means no limit.

```
star_He_mass_min_limit = 0
```

star_He_mass_max_limit = 0 ¶

Stop when star he3+he4 mass in Msun units is > this. <= 0 means no limit.

```
star_He_mass_max_limit = 0
```

star_C_mass_min_limit ¶

Stop when star c12 mass in Msun units is < this. <= 0 means no limit.

```
star_C_mass_min_limit = 0
```

star_C_mass_max_limit = 0 ¶

Stop when star c12 mass in Msun units is > this. <= 0 means no limit.

```
star_C_mass_max_limit = 0
```

envelope_mass_limit ¶

```
envelope_mass = star_mass - he_core_mass
```

Stop when `envelope_mass` drops below this limit, in Msun units.

```
envelope_mass_limit = 0
```

envelope_fraction_left_limit ¶

```
envelope_fraction_left = (star_mass - he_core_mass)/(initial_mass - he
```

```
envelope_fraction_left_limit = 0
```

xmstar_min_limit ¶

! xmstar = mstar - M_center stop when xmstar in grams is < this. <= 0 means no limit.

```
xmstar_min_limit = 0
```

xmstar_max_limit ¶

xmstar = mstar - M_center stop when xmstar in grams is > this. <= 0 means no limit.

```
xmstar_max_limit = 0
```

he_core_mass_limit ¶

stop when helium core reaches this mass, in Msun units

```
he_core_mass_limit = 1d99
```

c_core_mass_limit ¶

stop when carbon core reaches this mass, in Msun units

```
c_core_mass_limit = 1d99
```

o_core_mass_limit ¶

stop when oxygen core reaches this mass, in Msun units

```
o_core_mass_limit = 1d99
```

si_core_mass_limit ¶

stop when silicon core reaches this mass, in Msun units

```
si_core_mass_limit = 1d99
```

fe_core_mass_limit ¶

stop when iron core reaches this mass, in Msun units

```
fe_core_mass_limit = 1d99
```

neutron_rich_core_mass_limit ¶

stop when neutron rich core reaches this mass, in Msun units

```
neutron_rich_core_mass_limit = 1d99
```

he_layer_mass_lower_limit ¶

he layer mass is defined as `he_core_mass` - `c_core_mass` stop when `c_core_mass` > 0 and he layer mass < this limit (Msun units).

```
he_layer_mass_lower_limit = 0
```

abs_diff_lg_LH_lg_Ls_limit ¶

stop when `abs(lg_LH - lg_Ls) <= abs_diff_LH_Lsurf_limit` can be useful for deciding when pre-main sequence star has reached ZAMS set to negative value to disable

```
abs_diff_lg_LH_lg_Ls_limit = -1
```

Teff_upper_limit ¶

stop when Teff is greater than this limit.

```
Teff_upper_limit = 1d99
```

Teff_lower_limit ¶

stop when Teff is less than this limit.

```
Teff_lower_limit = -1d99
```

photosphere_r_upper_limit ¶

stop when `photosphere_r` is greater than this limit, in Rsun units

```
photosphere_r_upper_limit = 1d99
```

photosphere_r_lower_limit ¶

stop when `photosphere_r` is less than this limit, in Rsun units

```
photosphere_r_lower_limit = -1d99
```

photosphere_m_upper_limit ¶

stop when `photosphere_m` is greater than this limit, in Msun units

```
photosphere_m_upper_limit = 1d99
```

photosphere_m_lower_limit ¶

stop when `photosphere_m` is less than this limit, in Msun units

```
photosphere_m_lower_limit = -1d99
```

photosphere_m_sub_M_center_limit ¶

stop when `photosphere_m` is less than this limit above `M_center`, in Msun units

```
photosphere_m_sub_M_center_limit = -1d99
```

log_Teff_upper_limit ¶

stop when log10 of Teff is greater than this limit.

```
log_Teff_upper_limit = 1d99
```

log_Teff_lower_limit ¶

stop when log10 of Teff is less than this limit.

```
log_Teff_lower_limit = -1d99
```

log_Tsurf_upper_limit ¶

stop when log10 of T in outermost cell is greater than this limit.

```
log_Tsurf_upper_limit = 1d99
```

log_Tsurf_lower_limit ¶

stop when log10 of T in outermost cell is less than this limit.

```
log_Tsurf_lower_limit = -1d99
```

log_L_upper_limit ¶

stop when log10(total luminosity in Lsun units) is greater than this limit. in order to skip pre-ms, this limit only applies when `L_nuc` > 0.01*L

```
log_L_upper_limit = 1d99
```

log_L_lower_limit ¶

stop when log10(total luminosity in Lsun units) is less than this limit.

```
log_L_lower_limit = -1d99
```

log_g_upper_limit ¶

stop when log10(gravity at surface) is greater than this limit.

```
log_g_upper_limit = 1d99
```

log_g_lower_limit ¶

stop when log10(gravity at surface) is less than this limit.

```
log_g_lower_limit = -1d99
```

log_Psurf_upper_limit ¶

stop when log10 of surface pressure is greater than this limit.

```
log_Psurf_upper_limit = 1d99
```

log_Psurf_lower_limit ¶

stop when log10 of surface pressure is less than this limit.

```
log_Psurf_lower_limit = -1d99
```

log_Dsurf_upper_limit ¶

stop when log10 of surface density is greater than this limit.

```
log_Dsurf_upper_limit = 1d99
```

log_Dsurf_lower_limit ¶

stop when log10 of surface density is less than this limit.

```
log_Dsurf_lower_limit = -1d99
```

power_nuc_burn_upper_limit ¶

stop when total power from all nuclear reactions (in Lsun units) is > this.

```
power_nuc_burn_upper_limit = 1d99
```

power_h_burn_upper_limit ¶

stop when total power from hydrogen-consuming reactions (in Lsun units) is > this.

```
power_h_burn_upper_limit = 1d99
```

power_he_burn_upper_limit ¶

stop when total power from reactions burning helium (in Lsun units) is > this.

```
power_he_burn_upper_limit = 1d99
```

power_c_burn_upper_limit ¶

stop when total power from reactions burning carbon (in Lsun units) is > this

```
power_c_burn_upper_limit = 1d99
```

power_nuc_burn_lower_limit ¶

stop when total power from all nuclear reactions (in Lsun units) is < this.

```
power_nuc_burn_lower_limit = -1d99
```

power_h_burn_lower_limit ¶

stop when total power from hydrogen consuming reactions (in Lsun units) is < this.

```
power_h_burn_lower_limit = -1d99
```

power_he_burn_lower_limit ¶

stop when total power from reactions burning helium (in Lsun units) is < this.

```
power_he_burn_lower_limit = -1d99
```

power_c_burn_lower_limit ¶

stop when total power from reactions burning carbon (in Lsun units) is < this.

```
power_c_burn_lower_limit = -1d99
```

max_number_backups ¶

Stop if the number of backups exceeds this value. Ignore if < 0.

```
max_number_backups = -1
```

max_number_retries ¶

Stop if the number of retries exceeds this value. Ignore if < 0.

```
max_number_retries = -1
```

max_backups_in_a_row ¶

if do more than this many without a successful step, then terminate the run.

```
max_backups_in_a_row = 15
```

relax_max_number_backups ¶

Stop if the number of backups during a “relax” evolution exceeds this value. ignore if < 0

```
relax_max_number_backups = 100
```

relax_max_number_retries ¶

Stop if the number of retries during a “relax” evolution exceeds this value. ignore if < 0

```
relax_max_number_retries = 300
```

min_timestep_limit ¶

stop if need timestep smaller than this limit, in seconds

```
min_timestep_limit = 1d-6
```

logQ_limit ¶

logQ = logRho - 2*logT + 12. stop if logQ at any zone is larger than this limit. 5 is a reasonable limit for the current mesa/eos.

```
logQ_limit = 5d0
```

logQ_min_limit ¶

logQ = logRho - 2*logT + 12. stop if logQ at any zone is smaller than this limit. -10 is a reasonable limit for the current mesa/eos.

```
logQ_min_limit = -10d0
```

center_Ye_lower_limit ¶

stop if `center_ye` drops below this limit

```
center_Ye_lower_limit = -1
```

center_R_lower_limit ¶

stop if `R_center` drops below this limit (in cm)

```
center_R_lower_limit = -1
```

fe_core_infall_limit ¶

stop if max infall velocity at any location interior to `fe_core_mass`, in cm/s

```
fe_core_infall_limit = 1d8
```

non_fe_core_infall_limit ¶

stop if max infall velocity at any location interior to `he_core_mass`. in cm/s

```
non_fe_core_infall_limit = 1d99
```

v_div_csound_max_limit ¶

stop if any `v/csound` > this limit

```
v_div_csound_max_limit = 1d99
```

v_div_csound_surf_limit ¶

stop if `v_surf/csound_surf` > this limit

```
v_div_csound_surf_limit = 1d99
```

v_surf_div_v_kh_upper_limit ¶

stop if `abs(v_surf/v_kh)` > this limit, where `v_kh = photosphere_r/kh_timescale`

```
v_surf_div_v_kh_upper_limit = 1d99
```

v_surf_div_v_kh_lower_limit ¶

stop if `abs(v_surf/v_kh)` < this limit, where `v_kh = photosphere_r/kh_timescale`

```
v_surf_div_v_kh_lower_limit = -1d99
```

v_surf_div_v_esc_limit ¶

stop if `v_surf/v_esc` > this limit

```
v_surf_div_v_esc_limit = 1d99
```

v_surf_kms_limit ¶

stop if `v_surf` in km/s > this limit

```
v_surf_kms_limit = 1d99
```

Lnuc_div_L_zams_limit ¶

defines “near zams” – note: must also set `stop_near_zams`

```
Lnuc_div_L_zams_limit = 0.9d0
```

stop_near_zams ¶

if true, stop if Lnuc/L > `Lnuc_div_L_zams_limit`

```
stop_near_zams = .false.
```

Lnuc_div_L_upper_limit ¶

stop when Lnuc/L is greater than this limit.

```
Lnuc_div_L_upper_limit = 1d99
```

Lnuc_div_L_lower_limit ¶

stop when Lnuc/L is less than this limit.

```
Lnuc_div_L_lower_limit = -1d99
```

Pgas_div_P_limit ¶

criteria for stopping on Pgas/P

```
Pgas_div_P_limit = 0
```

Pgas_div_P_limit_max_q ¶

stop if Pgas/P < this limit at any location with `q <= Pgas_div_P_limit_max_q` values near unity skip the outer envelope

```
Pgas_div_P_limit_max_q = 0.95d0
```

peak_burn_vconv_div_cs_limit ¶

limits ratio of convection velocity to sound speed at location of peak `eps_nuc`

```
peak_burn_vconv_div_cs_limit = 1d99
```

omega_div_omega_crit_limit ¶

stop if omega/omega_crit is > this anywhere in star ignore if < 0

```
omega_div_omega_crit_limit = -1
```

delta_nu_lower_limit ¶

stop when asteroseismology `delta_nu` in micro Hz is < this. <= 0 means no limit.

```
delta_nu_lower_limit = 0
```

delta_nu_upper_limit ¶

stop when asteroseismology `delta_nu` in micro Hz is > this. <= 0 means no limit.

```
delta_nu_upper_limit = 0
```

shock_mass_upper_limit ¶

stop when shock_mass is > this. <= 0 means no limit.

```
shock_mass_upper_limit = -1
```

mach1_mass_upper_limit ¶

stop when outer location of mach 1 is > this. <= 0 means no limit.

```
mach1_mass_upper_limit = -1
```

delta_Pg_lower_limit ¶

stop when `delta_Pg` in micro Hz is < this. <= 0 means no limit.

```
delta_Pg_lower_limit = 0
```

delta_Pg_upper_limit ¶

stop when `delta_Pg` in micro Hz is > this. <= 0 means no limit.

```
delta_Pg_upper_limit = 0
```

stop_when_reach_this_cumulative_extra_heating ¶

(ignore if <= 0)

```
stop_when_reach_this_cumulative_extra_heating = 0d0
```

mixing parameters ¶

mixing_length_alpha ¶

The mixing length is this parameter times a local pressure scale height. To increase R vs. L, decrease `mixing_length_alpha`.

```
mixing_length_alpha = 2
```

remove_small_D_limit ¶

If MLT diffusion coeff D (cm^2/sec) is less than this limit, then set D to zero and change the point to `mixing_type == no_mixing`.

```
remove_small_D_limit = 1d-6
```

use_Ledoux_criterion ¶

a location in the model is Schwarzschild stable when `gradr < grada` it is Ledoux stable when `gradr < gradL`, where `gradL = grada + composition_gradient` note that these are the same when `composition_gradient = 0` so you can force the use of the Schwarzschild criterion by passing 0 for the `composition_gradient` argument to the mlt routine. that's what happens if you set the control "`use_Ledoux_criterion`" to false.

overshooting and rotational mixing are dealt with separately and are added after the MLT classifications are made.

```
use_Ledoux_criterion = .false.
```

num_cells_for_smooth_gradL_composition_term ¶

Number of cells on either side to use in weighted smoothing of `gradL_composition_term`. `gradL_composition_term` is set to the "raw" unsmoothed `brunt_B` and then optionally smoothed according `num_cells_for_smooth_gradL_composition_term`. In cases where the Ledoux criterion is used to evaluate the boundary for burning convective cores, you may need to set `num_cells_for_smooth_gradL_composition_term = 0` to avoid smoothing the stabilizing composition jump into the convection zone and unphysically causing it to shrink. See section 3.2 in Moore, K., & Garaud, P. 2016, APJ, 817, 54

```
num_cells_for_smooth_gradL_composition_term = 3
```

threshold_for_smooth_gradL_composition_term ¶

Threshold for weighted smoothing of `gradL_composition_term`. Only apply smoothing (controlled by `num_cells_for_smooth_gradL_composition_term`) for contiguous regions where |gradL| exceeds this threshold. Might be useful for preventing narrow composition jumps from being excessively broadened by smoothing

```
threshold_for_smooth_gradL_composition_term = 0
```

alpha_semiconvection ¶

Determines efficiency of semiconvective mixing. Semiconvection only applies if `use_Ledoux_criterion` is true.

```
alpha_semiconvection = 0
```

semiconvection_upper_limit_center_h1 ¶

Turn off semiconvection when `center_h1` > this limit. This let's you delay semiconvection until helium burning. E.g., you can do overshooting for core hydrogen burning, then switch to semiconvection after core h is gone.

```
semiconvection_upper_limit_center_h1 = 1d99
```

semiconvection_option ¶

- 'Langer_85 mixing; gradT = gradr' : uses Langer scheme for mixing but sets gradT = gradr
- 'Langer_85' : this calculates special gradT as well as doing mixing.

```
semiconvection_option = 'Langer_85 mixing; gradT = gradr'
```

thermohaline_coeff ¶

Determines efficiency of thermohaline mixing. was previously named `thermo_haline_coeff`. thermohaline mixing only applies if `use_Ledoux_criterion` is true.

```
thermohaline_coeff = 0
```

thermohaline_option ¶

determines which method to use for calculating thermohaline diffusion coef:

- `'Kippenhahn'` : use method of Kippenhahn, R., Ruschenplatt, G., & Thomas, H.-C. 1980, A&A, 91, 175.
- `'Traxler_Garaud_Stellmach_11'` : use method of Traxler, Garaud, & Stellmach, ApJ Letters, 728:L29 (2011).
- `'Brown_Garaud_Stellmach_13'` : use method of Brown, Garaud, & Stellmach, (2013). Recommends `thermohaline_coeff = 1`, but it can nevertheless be changed.

```
thermohaline_option = 'Kippenhahn'
```

alt_scale_height_flag ¶

If false, then stick to the usual definition – P/(g*rho). If true, use min of the usual and sound speed * hydro time scale, sqrt(P/G)/rho.

```
alt_scale_height_flag = .true.
```

mlt_use_rotation_correction ¶

When doing rotation, multiply `grad_rad` by `ft_rot/ft_rot` if this flag is true.

```
mlt_use_rotation_correction = .true.
```

MLT_option ¶

Options are:

- 'none' : just give radiative values with no mixing.
- 'Cox' : MLT as developed in Cox & Giuli 1968, Chapter 14.
- 'ML1' : Bohm-Vitense 1958

- ‘ML2’ : Bohm and Cassinelli 1971
- ‘Mihalas’ : Mihalas 1978, Kurucz 1979
- ‘Henyey’ : Henyey, Vardya, and Bodenheimer 1965

‘Cox’ option assumes optically thick material. The other options are various ways of extending to include optically thin material.

```
MLT_option = 'Cox'
```

Henyey_MLT_y_param ¶

Henyey_MLT_nu_param ¶

Values of the f1..f4 coefficients are taken from Table 1 of Ludwig et al. 1999, A&A, 346, 111 with the following exception: their value of f3 for Henyey convection is `f4/8` when it should be `8*f4`, i.e., `f3=32*pi**2/3` and `f4=4*pi**2/3`. f3 and f4 are related to the henyey y parameter, so for the ‘Henyey’ case they are set based on the value of `Henyey_y_param`.

```
Henyey_MLT_y_param = 0.33333333d0
Henyey_MLT_nu_param = 8
```

make_gradr_sticky_in_newton_iters ¶

if true, then location that becomes radiative during newton iterations, stays radiative for rest of the newton iterations. to avoid flip-flopping between radiative and convective.

```
make_gradr_sticky_in_newton_iters = .false.
```

no_MLT_below_shock ¶

if true, then no MLT below an outward going shock (just radiative).

```
no_MLT_below_shock = .false.
```

no_MLT_below_T_max ¶

if true, then no MLT below location of max T (just radiative).

```
no_MLT_below_T_max = .false.
```

T_mix_limit ¶

If there is any convection in surface zones with `T < T_mix_limit`, then extend the innermost such convective region outward all the way to the surface. For example,

- `T_mix_limit <= 0` means omit this operation.
- `T_mix_limit = 1d5` will effectively make the star convective down to the He++ region.

units in Kelvin

```
T_mix_limit = 0
```

conv_dP_term_factor ¶

Set to 0 to turn off effect of pressure from convective turbulence. The convective turbulence factor is based on Cox&Giuli (14.69) Multiplier for `conv_dP_term` P is increased by factor `(1 + conv_dP_term)` by inclusion of convective turbulence.

```
conv_dP_term_factor = 0
```

mlt_gradT_fraction ¶

let f := `mlt_gradT_fraction` if f is >= 0 and <= 1, then gradT from mlt is replaced by `f*grada_at_face(k) + (1-f)*gradr(k)` see also the vector control `adjust_mlt_gradT_fraction` for fine grain control

```
mlt_gradT_fraction = -1
```

okay_to_reduce_gradT_excess ¶

`gradT_excess` = `gradT_sub_grada` = superadiabaticity.

Inefficient convection => large gradT excess and steep T gradient to enhance radiative transport. Reduce gradT excess by making gradT closer to adiabatic gradient. If true, code is allowed to adjust gradT to boost efficiency of energy transport See `gradT_excess_f1`, `gradT_excess_f2`, and `gradT_excess_age_fraction` below.

```
okay_to_reduce_gradT_excess = .false.
```

gradT_excess_f1 ¶

gradT_excess_f2 ¶

These are for calculation of efficiency boosted gradT.

```
gradT_excess_f1 = 1d-4
gradT_excess_f2 = 1d-3
```

gradT_excess_age_fraction ¶

These are for calculation of efficiency boosted gradT. Fraction of old to mix with new to get next.

```
gradT_excess_age_fraction = 0.9d0
```

gradT_excess_max_change ¶

These are for calculation of efficiency boosted gradT. Maximum change allowed in one timestep for `gradT_excess_alpha`. Ignored if negative.

```
gradT_excess_max_change = -1d0
```

gradT_excess_lambda1 ¶

gradT_excess_beta1 ¶

In some situations you might want to force alfa = 1. You can do that by setting `gradT_excess_lambda1 < 0`. The following are for the normal calculation of `gradT_excess_alfa`

```
gradT_excess_lambda1 = 1.0d0
gradT_excess_beta1 = 0.35d0
```

gradT_excess_lambda2 ¶

gradT_excess_beta2 ¶

The following are for the normal calculation of `gradT_excess_alfa`.

```
gradT_excess_lambda2 = 0.5d0
gradT_excess_beta2 = 0.25d0
```

gradT_excess_dlambda ¶

gradT_excess_dbeta ¶

The following are for the normal calculation of `gradT_excess_alfa`.

```
gradT_excess_dlambda = 0.1d0
gradT_excess_dbeta = 0.1d0
```

gradT_excess_max_center_h1 ¶

No boost if center H1 > this limit.

```
gradT_excess_max_center_h1 = 1d0
```

gradT_excess_min_center_he4 ¶

No boost if center He4 < this limit.

```
gradT_excess_min_center_he4 = 0d0
```

gradT_excess_max_logT ¶

No local boost if local logT > this limit.

```
gradT_excess_max_logT = 8
```

gradT_excess_min_log_tau_full_on ¶

gradT_excess_max_log_tau_full_off ¶

No local boost if local `log_tau < gradT_excess_max_log_tau_full_off`. Reduced local boost if local `log_tau < gradT_excess_min_log_tau_full_on`.

```
gradT_excess_min_log_tau_full_on = -99
gradT_excess_max_log_tau_full_off = -99
```

smooth_gradT ¶

use_grada_for_smooth_gradT ¶

gradT_smooth_low ¶

gradT_smooth_mid ¶

gradT_smooth_high ¶

gradT_smooth_factor ¶

EXPERIMENTAL: soften gradT at the boundaries of convective zones to help convergence

```
smooth_gradT = .false.
use_grada_for_smooth_gradT = .false.
gradT_smooth_low = -0.005d0
gradT_smooth_mid = 0d0
gradT_smooth_high = 0.01d0
gradT_smooth_factor = 1.0d0
```

max_logT_for_mlt ¶

No mlt at cell if local logT > this limit.

```
max_logT_for_mlt = 99
```

overshooting ¶

Overshooting depends on the classification of the convective zone and can be different at the top and the bottom of the zone.

min_overshoot_q ¶

Overshooting is only allowed at locations with mass `m >= min_overshoot_q * mstar`. E.g., if `min_overshoot_q = 0.1`, then only the outer 90% by mass can have overshooting. This provides a simple way of suppressing bogus center overshooting in which a small convective region at the core can produce excessively large overshooting because of a large pressure scale height at the center.

```
min_overshoot_q = 1d-3
```

D_mix_ov_limit ¶

Overshooting shuts off when the exponential decay has dropped the diffusion coefficient to this level.

```
D_mix_ov_limit = 1d2
```

max_brunt_B_for_overshoot ¶

Terminate overshoot region when encounter stabilizing composition gradient where (unsmoothed) `brunt_B` is greater than this limit. (<= 0 means ignore this limit) note: both `brunt_B` and `gradL_composition_term` come from `unsmoothed_brunt_B` and differ only in optional smoothing. (see `num_cells_for_smooth_brunt_B` and `num_cells_for_smooth_gradL_composition_term`).

```
max_brunt_B_for_overshoot = 0
```

Parameters for exponential diffusive overshoot are described in the paper by Falk Herwig, “The evolution of AGB stars with convective overshoot”, A&A, 360, 952-968 (2000).

NOTE: In addition to giving these ‘f’ parameters non-zero values, you should also check the settings for `mass_for_overshoot_full_on` and `mass_for_overshoot_full_off`.

The switch from convective mixing to overshooting happens at a distance f0*Hp into the convection zone from the estimated location where `grad_ad == grad_rad`, where Hp is the pressure scale height at that location. A value <= 0 for f0 is a mistake – you are required to set f0 as well as f. take a look at the following from an email concerning this:

Overshooting works by taking the diffusion mixing coefficient at the edge of the convection zone and extending it beyond the zone. But – and here’s the issue – at the exact edge of the zone the mixing coefficient goes to 0. So we don’t want that. Instead we want the value of the mixing coeff NEAR the edge, but not AT the edge. The “f0” parameter determines the exact meaning of “near” for this. It tells the code how far back into the zone to go in terms of scale height. The overshooting actually begins at the location determined by f0 back into the convection zone rather than at the edge where the diffusion coeff is ill-defined. So, for example, if you want overshooting of 0.2 scale heights beyond the normal edge, you might want to back up 0.05 scale heights to get the diffusion coeff from near the edge and then go out by 0.25 scale heights from there to reach 0.2 Hp beyond the old boundary. In the inlist this would mean setting the “f0” to 0.05 and the “f” to 0.25.

There is no default value for f0; if you set f > 0 then you must get f0 > 0 as well.

overshoot_alpha ¶

The value of Hp for overshooting is limited to the radial thickness of the convection zone divided by `overshoot_alpha`. only used when > 0. if <= 0, then use `mixing_length_alpha` instead.

```
overshoot_alpha = -1
```

limit_overshoot_Hp_using_size_of_convection_zone ¶

if false, allow large distance of overshoot for small convective zones.

```
limit_overshoot_Hp_using_size_of_convection_zone = .true.
```

overshoot_f_above_nonburn_core ¶

overshoot_f0_above_nonburn_core ¶

overshoot_f_above_nonburn_shell ¶

overshoot_f0_above_nonburn_shell ¶

overshoot_f_below_nonburn_shell ¶

overshoot_f0_below_nonburn_shell ¶

For nonburning regions.

```
overshoot_f_above_nonburn_core = 0
overshoot_f0_above_nonburn_core = -1
overshoot_f_above_nonburn_shell = 0
overshoot_f0_above_nonburn_shell = -1
overshoot_f_below_nonburn_shell = 0
overshoot_f0_below_nonburn_shell = -1
```

overshoot_f_above_burn_h_core ¶

overshoot_f0_above_burn_h_core ¶

overshoot_f_above_burn_h_shell ¶

overshoot_f0_above_burn_h_shell ¶

overshoot_f_below_burn_h_shell ¶

overshoot_f0_below_burn_h_shell ¶

For hydrogen burning regions.

```
overshoot_f_above_burn_h_core = 0
overshoot_f0_above_burn_h_core = -1
overshoot_f_above_burn_h_shell = 0
overshoot_f0_above_burn_h_shell = -1
overshoot_f_below_burn_h_shell = 0
overshoot_f0_below_burn_h_shell = -1
```

overshoot_f_above_burn_he_core ¶

overshoot_f0_above_burn_he_core ¶

overshoot_f_above_burn_he_shell ¶

overshoot_f0_above_burn_he_shell ¶

overshoot_f_below_burn_he_shell ¶

overshoot_f0_below_burn_he_shell ¶

For helium burning regions.

```
overshoot_f_above_burn_he_core = 0
overshoot_f0_above_burn_he_core = -1
overshoot_f_above_burn_he_shell = 0
overshoot_f0_above_burn_he_shell = -1
overshoot_f_below_burn_he_shell = 0
overshoot_f0_below_burn_he_shell = -1
```

overshoot_f_above_burn_z_core ¶

overshoot_f0_above_burn_z_core ¶

overshoot_f_above_burn_z_shell ¶

overshoot_f0_above_burn_z_shell ¶

overshoot_f_below_burn_z_shell ¶

overshoot_f0_below_burn_z_shell ¶

For metals burning regions.

```
overshoot_f_above_burn_z_core = 0
overshoot_f0_above_burn_z_core = -1
overshoot_f_above_burn_z_shell = 0
overshoot_f0_above_burn_z_shell = -1
overshoot_f_below_burn_z_shell = 0
overshoot_f0_below_burn_z_shell = -1
```

overshoot_below_noburn_shell_factor ¶

Multiply `overshoot_f_below_nonburn_shell` by this factor only during dredge up phase of AGB thermal pulse.

```
overshoot_below_noburn_shell_factor = 1
```

Optional step function for overshooting. This can be used simultaneously with exponential overshooting. When using step overshoot, you must set overshoot_f0 as well as f.

A convective region is considered a shell if it doesn't reach the center.

```
step_overshoot_f_above_nonburn_core = 0
step_overshoot_f_above_nonburn_shell = 0
step_overshoot_f_below_nonburn_shell = 0
```

```
step_overshoot_f_above_burn_h_core = 0
step_overshoot_f_above_burn_h_shell = 0
step_overshoot_f_below_burn_h_shell = 0
```

```
step_overshoot_f_above_burn_he_core = 0
step_overshoot_f_above_burn_he_shell = 0
step_overshoot_f_below_burn_he_shell = 0
```

```
step_overshoot_f_above_burn_z_core = 0
step_overshoot_f_above_burn_z_shell = 0
step_overshoot_f_below_burn_z_shell = 0
```

step_overshoot_D ¶

step_overshoot_D0_coeff ¶

As above, `f0*Hp` determines r0 where switch from convection to overshooting. Overshooting extends a distance `step_f*Hp0` from r0 with constant diffusion coeff `D = step_D + step_D0_coeff*D0` where D0 = diffusion coefficient D at point r0.

```
step_overshoot_D = 0
step_overshoot_D0_coeff = 1
```

mass_for_overshoot_full_on ¶

You can specify a range of star masses over which overshooting above H burning zones is gradually enabled. Do specified overshooting above H burning zone if `star_mass` >= this (Msun).

```
mass_for_overshoot_full_on = 0
```

mass_for_overshoot_full_off ¶

You can specify a range of star masses over which overshooting above H burning zones is gradually enabled. No overshooting above H burning zone if `star_mass` <= this (Msun).

```
mass_for_overshoot_full_off = 0
```

DUP_varcontrol_factor ¶

```
DUP_varcontrol_factor = 1d0
```

max_DUP_counter ¶

For deciding when to terminate use of `overshoot_below_noburn_shell_factor`.

```
max_DUP_counter = 200
```

ovr_below_burn_he_shell_factor ¶

Multiply `overshoot_f_below_burn_he_shell` by this factor after the first AGB thermal pulse.

```
ovr_below_burn_he_shell_factor = 1
```

overshoot_D2 ¶

overshoot_f2 ¶

optional 2nd scale length for exponential overshooting

`f0*Hp` determines location r0 where we switch from convection to overshooting. Let D0 = diffusion coefficient D at point r0. Let Hp0 = the scale height at r0. In the standard version of exponential overshooting, there is a single length scale = `f*Hp0` and at a distance dr from r0, `D(dr) = D0*exp(-2*dr/(f*Hp0))`.

In the extended version there is a second length scale = `f2*Hp0`. The second length scale takes effect for distances dr > dr2 where dr2 is defined by `D2 = D0*exp(-2*dr2/(f*Hp0))`.

- for dr <= dr2, D(dr) = `D0*exp(-2*dr/(f*Hp0))`
- for dr > dr2, D(dr) = `D2*exp(-2*(dr-dr2)/(f2*Hp0))` = `D0*exp(-2*dr2/(f*Hp0))*exp(-2*(dr-dr2)/(f2*Hp0))` = `D0*exp(-2*(dr2/(f*Hp0) + (dr-dr2)/(f2*Hp0)))`

```
overshoot_D2_above_nonburn = -1d0
overshoot_D2_below_nonburn = -1d0
```

```
overshoot_D2_above_burn_h = -1d0
overshoot_D2_below_burn_h = -1d0
```

```
overshoot_D2_above_burn_he = -1d0
overshoot_D2_below_burn_he = -1d0
```

```
overshoot_D2_above_burn_z = -1d0
overshoot_D2_below_burn_z = -1d0
```

```
overshoot_f2_above_nonburn = 1d0
overshoot_f2_below_nonburn = 1d0
```

```
overshoot_f2_above_burn_h = 1d0
overshoot_f2_below_burn_h = 1d0
```

```
overshoot_f2_above_burn_he = 1d0
overshoot_f2_below_burn_he = 1d0
```

```
overshoot_f2_above_burn_z = 1d0
overshoot_f2_below_burn_z = 1d0
```

RGB_to_AGB_cbm_switch ¶

If center hydrogen abundance is < 0.01 and center helium abundance by mass is less than `RGB_to_AGB_cbm_switch`, then system will include `overshoot_D2` and `overshoot_f2` parameters to describe convective boundary mixing during the AGB phase if set to positive values. See Battino et al. 2016, APJ, 827:30
"Application of a theory and simulation-based convective boundary mixing model for AGB star evolution and nucleosynthesis" `RGB_to_AGB_cbm_switch = 1d-4` is used in the paper above.

```
RGB_to_AGB_cbm_switch = -1
```

Predictive mixing ¶

Predictive mixing is an approach for expanding convective boundaries until gradr = grada on the convective side of the boundary (as required by the criterion that the convective velocity and luminosity vanish at the boundary). It is discussed in detail in Paxton et al. 2018, ApJ, in press: "Modules for Experiments in Stellar Astrophysics (MESA): Convective boundaries, element diffusion, and massive star explosions"

Predictive mixing is controlled by specifying a set of parameters, which combines matching criteria (determining which boundaries to apply the predictive mixing to) together with values (determining how the predictive mixing should operate at those boundaries). Up to `NUM_PREDICTIVE_PARAM_SETS` of these parameter sets can be defined (see `star_def.inc` for value).

predictive_mix ¶

Set to .true. to enable this set of parameters

```
predictive_mix(1) = .false.
```

predictive_zone_type ¶

Matching criterion for the type of the convection zone. Possible values are `burn_H` (hydrogen burning), `burn_He` (helium burning), `burn_Z` (metal burning), `nonburn` (no burning) or `any` (which matches any type of zone).

```
predictive_zone_type(1) = ''
```

predictive_zone_loc ¶

Matching criterion for the location of the convection zone. Possible values are `core` (the core convection zone), `shell` (a convective shell), `surf` (the surface convection zone) or `any` (which matches any location).

```
predictive_zone_loc(1) = ''
```

predictive_bdy_loc ¶

Matching criterion for the location of the convective boundary. Possible values are `top` (the top of the convection zone), `bottom` (the bottom of the convection zone) or `any` (which matches any location).

```
predictive_bdy_loc(1) = ''
```

predictive_bdy_q_min ¶

Matching criterion for the minimum fractional mass coordinate of the convective boundary

```
predictive_bdy_q_min(1) = 0d0
```

predictive_bdy_q_max ¶

Matching criterion for the maximum fractional mass coordinate of the convective boundary

```
predictive_bdy_q_max(1) = 1d0
```

predictive_superad_thresh ¶

Threshold for minimum superadiabaticity in the predictive mixing scheme; boundary expansion stops when gradr/grada-1 drops below this threshold. Default value is usually good for main-sequence evolution; for

core He-burning, set to 0.005, 0.01 or larger to prevent splitting of the core convection zone and/or core breathing pulses.

```
predictive_superad_thresh(1) = 0d0
```

predictive_avoid_reversal ¶

Species to monitor for reversals in abundance evolution. If this is set to the name of a species, then the predictive mixing scheme will try to avoid causing reversals in the abundance of that species (e.g., changing the abundance evolution from decreasing to increasing). Set to 'he4' during core He-burning to prevent splitting of the core convection zone and/or core breathing pulses.

```
predictive_avoid_reversal(1) = ''
```

predictive_limit_ingestion ¶

predictive_ingestion_factor ¶

Limit the rate of ingestion of a species, following the prescription given in equation (2) of Constantino, Campbell & Lattanzio (2017, MNRAS, 472, 4900). The control `predictive_limit_ingestion` specifies which species to limit, and the control `predictive_ingestion_factor` gives the multiplying factor. Setting this factor to 5/12 is the same as choosing alpha_i = 1 in their equation (2).

```
predictive_limit_ingestion(1) = ''
predictive_ingestion_factor(1) = 0d0
```

Self-Driving Overshoot & Semiconvection (S-DOS) mixing ¶

```
do_sdos_mix = .false.
sdos_avoid_reversal = .false.
```

New overshooting ¶

```
overshoot_new = .false.
overshoot_f = 0d0
overshoot_f0 = 0d0
overshoot_f2 = 0d0
overshoot_D0 = 0d0
overshoot_D2 = 0d0
overshoot_Delta0 = 1d0
overshoot_c1 = 100d0
overshoot_c2 = 90d0
overshoot_mass_full_on = 0d0
overshoot_mass_full_off = 0d0
overshoot_scheme = ''
overshoot_zone_type = ''
overshoot_zone_loc = ''
overshoot_bdy_loc = ''
```

```
overshoot_D_min = 1d2
overshoot_brunt_B_max = 0d0
```

turbulence ¶

RTI_max_time_full_off ¶

RTI_min_time_full_on ¶

```
RTI_max_time_full_off = 0d0
RTI_min_time_full_on = 0d0
```

RTI_smooth_mass ¶

RTI_smooth_iterations ¶

RTI_smooth_fraction ¶

smoothing for `dPdr_dRhodr_info` done at start of step

```
RTI_smooth_mass = 0d0
RTI_smooth_iterations = 0
RTI_smooth_fraction = 1d0
```

alpha_RTI_diffusion_factor ¶

dudt_RTI_diffusion_factor ¶

dedt_RTI_diffusion_factor ¶

dlnddt_RTI_diffusion_factor ¶

composition_RTI_diffusion_factor ¶

max_M_RTI_factors_full_on ¶

min_M_RTI_factors_full_off ¶

```
alpha_RTI_diffusion_factor = 1d0
dudt_RTI_diffusion_factor = 1d0
dedt_RTI_diffusion_factor = 1d0
dlnddt_RTI_diffusion_factor = 1d0
composition_RTI_diffusion_factor = 1d0
max_M_RTI_factors_full_on = 1d99
min_M_RTI_factors_full_off = 1d99
```

alpha_RTI_src_max_q ¶

alpha_RTI_src_min_q ¶

option to set `alpha_RTI` source term to zero when cell q out of bounds. to turn off RTI near surface or center

```
alpha_RTI_src_max_q = 1d0
alpha_RTI_src_min_q = 0d0
```

alpha_RTI_src_min_v_div_cs ¶

option to set alpha_RTI source term to zero when v/cs < this min. e.g. to filter out false sources ahead of shock

```
alpha_RTI_src_min_v_div_cs = 1d0
```

radiation_turbulence_coeff ¶

To counter depletion of h and metals in outer envelope of stars with M > 1.4 Msun. Morel, P., and Thevenin, F., Atomic diffusion in stellar models of type earlier than G., A&A, 390:611-620 (2002)

```
D = radiation_turbulence_coeff * 4*crad*T^4/(15*clight*opacity*rho^2)
```

1 is reasonable value for this coefficient

```
radiation_turbulence_coeff = 0
```

turbulent_diffusion_D0 ¶

Turbulent diffusion below outer convection zone. Similar effect to overshooting. Proffitt, C.R., and Michaud, G., GRAVITATIONAL SETTLING IN SOLAR MODELS, ApJ, 380:238-290, 1991. e.g., 8000 cm^2 s^-1

```
turbulent_diffusion_D0 = 0
```

turbulent_diffusion_rho_max ¶

Only have turbulent diffusion if rho < this. Diffusion coef `D = D0*(rho/rho_base_cz)^-3`, but only if `rho_base_cz <= turbulent_diffusion_rho_max`.

```
turbulent_diffusion_rho_max = 1d99
```

turbulent_diffusion_Dmin ¶

Only set turbulent diffusion if it gives D >= this.

```
turbulent_diffusion_Dmin = 1d1
```

mixing misc ¶

such as smoothing and editing of diffusion coefficients

mix_factor ¶

Mixing coefficients are multiplied by this factor. The `mix_factor` is applied in subroutine `get_convection_sigmas` in `star/private/mix_info.f90` – the lagrangian diffusion coefficient sigma(k) at cell boundary k is set to `mix_factor*D*(4*pi*r(k)^2*rho_face(k))^2` Note that the value of D is not changed – it is just used as a term in calculating sigma.

```
mix_factor = 1
```

min_dt_for_increases_in_convection_velocity ¶

convective velocities are not increased if dt < this value (in seconds)

```
min_dt_for_increases_in_convection_velocity = -1d0
```

max_conv_vel_div_csound ¶

convective velocities are limited to local sound speed times this factor

```
max_conv_vel_div_csound = 1d99
```

max_v_div_cs_for_convection ¶

disable convection for locations with abs(v)/cs > this limit

```
max_v_div_cs_for_convection = 1d99
```

max_abs_du_div_cs_for_convection ¶

main purpose is to force radiative in shock face

```
max_abs_du_div_cs_for_convection = 0.03
```

min_T_for_acceleration_limited_conv_velocity ¶

Acceleration limiting based on Wood 1974 and Arnett 1969. Wood, P.R., ApJ, 190:609-630, 1974. (Appendix V, eqns 1-3) Arnett, W.D., 1969, Ap. and Space Sci, 5, 180.

```
min_T_for_acceleration_limited_conv_velocity = 99e9
```

max_T_for_acceleration_limited_conv_velocity ¶

```
max_T_for_acceleration_limited_conv_velocity = 99e9
```

mlt_accel_g_theta ¶

use this (if > 0) for limiting the acceleration of convection velocities.

```
mlt_accel_g_theta = -1
```

prune_bad_cz_min_Hp_height ¶

Lower limit on radial extent of cz (<= 0 to disable). Remove tiny convection zones unless have strong nuclear burning i.e., remove if `size < prune_bad_cz_min_Hp_height` .and. `max_log_eps < prune_bad_cz_min_log_eps_nuc`.

```
prune_bad_cz_min_Hp_height = 0
```

prune_bad_cz_min_log_eps_nuc ¶

Lower limit on max log eps nuc in cz. In units of average pressure scale height at top and bottom of region. This allows emergence of very small cz at site of he core flash, for example.

```
prune_bad_cz_min_log_eps_nuc = -99
```

redo_conv_for_dr_lt_mixing_length ¶

Check for small convection zones with total height less than mixing length and redo with reduced `mixing_length_alpha` to make `mixing_length <= dr`.

```
redo_conv_for_dr_lt_mixing_length = .false.
```

limit_mixing_length_by_dist_to_bdy ¶

reduce local value of mixing length alpha if necessary in order to make mixing length <= distance to convective boundary times this value only applies when value is > 0 setting this value = 1 implements the restriction that near a convective boundary, the mixing length doesn't exceed the distance to the boundary. Peter Eggleton, "Composition Changes during Stellar Evolution", MNRAS 156, 361-376, 1972.

WARNING: I've seen problems with 25M before He core burn when using this. bp.

```
limit_mixing_length_by_dist_to_bdy = 0
```

conv_bdy_mix_softening_f0 ¶

conv_bdy_mix_softening_f ¶

conv_bdy_mix_softening_min_D_mix ¶

These controls cause the convective mixing coefficient to drop off “softly” near the boundary of the convective zone – i.e, they prevent situations where the mixing coefficient drops from 10^10 or more to zero in a distance covered by only one or two cells as can happen at jumps in composition. Such a sharp edge is no problem when it is not adjacent to a convective region. But when it shows up at a convective boundary, it is problematic. It may not be physical, and it is certainly bad news numerically. This has been discussed as early as 1970’s – see for example, Peter Eggleton, “Composition Changes during Stellar Evolution”, MNRAS 156, 361-376, 1972.

The implementation of softening at convective boundaries is like overshooting but the distances are typically smaller by an order of magnitude or more. Also, the softening is primarily inside the convective region rather than penetrating strongly into the area beyond. This is done by backing up from the boundary into the convection region to start the softening of the mixing coefficient so that most of the effect takes place before reaching the exterior of the region. The softening extends a short way into the exterior with a decreasing value of mixing. For example, it might start a distance of `0.003*Hp` into the convective region (Hp = pressure scale height at the boundary), and then project a mixing coefficient outward from there decreasing exponentially with distance scale of `0.001*Hp`. For those numbers, there are 3 e-foldings before reaching the boundary, so most of the drop has happened inside the convective region. The strength of the mixing then continues to drop exponentially until it reaches some given limit.

The effect of this will be to soften the jump in abundances immediately adjacent to the convective region. That will in turn soften the jump in opacity and the corresponding jump in `grad_rad` so that there will not be a large jump from a convective point with `grad_rad >> grad_ad` to a neighboring non-convective point with `grad_rad << grad_ad`.

```
conv_bdy_mix_softening_f0 = 0
conv_bdy_mix_softening_f = 0
conv_bdy_mix_softening_min_D_mix = 0
```

smooth_convective_bdy ¶

This is an option to smooth composition gradients in newly non-convective regions trailing behind a retreating convection zone. This effectively erases (most) of the stair-casing that happens without it. But you should be aware that the smoothing process does not conserve species mass – e.g., if have retreating He burning core below H shell, then the smoothing will convert some H into He in the newly non-convective region (this can be hand waved away as modeling partial burning of those regions during the substep period before the convection had retreated past the location).

set this true to have the stair-casing removed at the price of some changes in abundances.

```
smooth_convective_bdy = .true.
```

max_dR_div_Hp_for_smooth ¶

Don’t smooth across a newly nonconvective region larger than this limit where dR is radial thickness of region and Hp is min pressure scale height in region.

```
max_dR_div_Hp_for_smooth = 10
```

max_delta_limit_for_smooth ¶

Don’t smooth across a newly nonconvective region where any mass fraction changes by more than this limit.

```
max_delta_limit_for_smooth = 0.1d0
```

remove_mixing_glitches ¶

If true, then okay to remove gaps and singletons.

```
remove_mixing_glitches = .true.
```

glitches ¶

The following controls are for different kinds of “glitches” that can be removed.

okay_to_remove_mixing_singleton ¶

If true, remove singetons.

```
okay_to_remove_mixing_singleton = .true.
```

clip_D_limit ¶

Zero mixing diffusion coeffs that are smaller than this.

```
clip_D_limit = 0
```

min_convective_gap ¶

Close gap between convective regions if smaller than this (< 0 means skip this). Gap measured radially in units of pressure scale height.

```
min_convective_gap = -1
```

min_thermohaline_gap ¶

Close gap between thermohaline mixing regions if smaller than this (< 0 means skip this). Gap measured radially in units of pressure scale height.

```
min_thermohaline_gap = -1
```

min_thermohaline_dropout ¶

max_dropout_gradL_sub_grada ¶

If find radiative region embedded in thermohaline, and `max(gradL - grada)` in region is everywhere `< max_dropout_gradL_sub_grada` and region height is `< min_thermohaline_dropout` then convert the region to thermohaline. `min_thermohaline_dropout <= 0` disables.

```
min_thermohaline_dropout = -1
max_dropout_gradL_sub_grada = 1d-3
```

min_semiconvection_gap ¶

Close gap between semiconvective mixing regions if smaller than this (< 0 means skip this). Gap measured radially in units of pressure scale height.

```
min_semiconvection_gap = -1
```

remove_embedded_semiconvection ¶

If have a semiconvection region bounded on each side by convection, convert it to be convective too.

```
remove_embedded_semiconvection = .false.
```

set_min_D_mix ¶

mass_lower_limit_for_min_D_mix ¶

mass_upper_limit_for_min_D_mix ¶

min_D_mix ¶

`D_mix` will be at least this large if `set_min_D_mix` is true. doesn't apply for mass < lower limit or mass > upper limit.

```
set_min_D_mix = .false.
mass_lower_limit_for_min_D_mix = 0d0
mass_upper_limit_for_min_D_mix = 1d99
min_D_mix = 1d3
```

set_min_D_mix_in_H_He ¶

min_D_mix_in_H_He ¶

`D_mix` will be at least this large in regions where max mass fractions of H and He add to more that 0.5 if `set_min_D_mix_in_H_He` is true.

```
set_min_D_mix_in_H_He = .false.
min_D_mix_in_H_He = 1d3
```

set_min_D_mix_below_Tmax ¶

min_D_mix_below_Tmax ¶

`D_mix` will be at least this large for cells below location of max temperature if `set_min_D_mix_below_Tmax` is true.

```
set_min_D_mix_below_Tmax = .false.
min_D_mix_below_Tmax = 1d3
```

min_center_Ye_for_min_D_mix ¶

`min_D_mix` is only used when `center_ye >= this` i.e., when `center_ye` drops below this, `min_D_mix = 0`.

```
min_center_Ye_for_min_D_mix = 0.47d0
```

smooth_outer_xa_big ¶

smooth_outer_xa_small ¶

Soften composition jumps in outer layers. If `smooth_outer_xa_big` and `smooth_outer_xa_small` are bigger than 0, then starting from the outermost grid point, homogeneously mix a region of size `smooth_outer_xa_small` (in solar masses), and proceed inwards, linearly reducing the size of the homogeneously mixed region in such a way that it becomes zero. After going `smooth_outer_xa_big` solar masses in. In this way, the outer `smooth_outer_xa_big` solar masses are “cleaned” of composition jumps.

```
smooth_outer_xa_big = -1d0
smooth_outer_xa_small = -1d0
```

rotation controls ¶

In the following “am” stands for “angular momentum”.

the mesa implementation of rotation closely follows these papers:

- Heger, Langer, & Woosley, ApJ, 528, 368. 2000
- Heger, Woosley, & Spruit, ApJ, 626, 350. 2005

- `D_DSI` = dynamical shear instability
- `D_SH` = Solberg-Hoiland
- `D_SSI` = secular shear instability
- `D_ES` = Eddington-Sweet circulation

- D_GSF = Goldreich-Schubert-Fricke
- D_ST = Spruit-Tayler dynamo

skip_rotation_in_convection_zones ¶

if true, then set rotational diffusion coefficients to 0 in convective regions. This applies both for material mixing and diffusion of angular momentum.

```
skip_rotation_in_convection_zones = .false.
```

am_D_mix_factor ¶

Rotation and mixing of material. D_mix = diffusion coefficient for mixing of material. It is sum of non-rotational and rotational components. The rotational part is multiplied by this factor.

```
D_mix = D_mix_non_rotation + f*am_D_mix_factor*(
    D_DSI_factor * D_DSI +
    D_SH_factor  * D_SH +
    D_SSI_factor * D_SSI +
    D_ES_factor  * D_ES +
    D_GSF_factor * D_GSF +
    D_ST_factor  * D_ST)

f = 1  when logT <= D_mix_rotation_max_logT_full_on = full_on
  = 0  when logT >= D_mix_rotation_max_logT_full_on = full_off
  = (log(T)-full_on)/(full_off-full_on) else
```

note that for regions with brunt N^2 < 0, we set Richardson number to 1 which is > Ri_critical and therefore turns off DSI and SSI

according to Heger et al 2000 : 1/30d0 by default : 0

```
am_D_mix_factor = 0
```

am_nu_factor ¶

am_nu_non_rotation_factor ¶

diffusion of angular momentum

am_nu = diffusion coefficient for angular momentum

```
am_nu_non_rot = am_nu_factor*am_nu_non_rotation_factor*D_mix_non_rotat
am_nu_rot = am_nu_factor*(
    am_nu_visc_factor* D_visc +
    am_nu_DSI_factor * D_DSI +
    am_nu_SH_factor  * D_SH +
    am_nu_SSI_factor * D_SSI +
    am_nu_ES_factor  * D_ES +
```

```
     am_nu_GSF_factor * D_GSF +
     am_nu_ST_factor  * nu_ST)
am_nu = am_nu_non_rot + am_nu_rot
```

Note that for regions with brunt N^2 < 0, we set Richardson number to 1 which is > Ri_critical and therefore turns off DSI and SSI.

see also `star_job` controls for `am_nu_rot_flag`

```
am_nu_factor = 1
am_nu_non_rotation_factor = 1
```

am_nu_DSI_factor ¶

< 0 means use `D_DSI_factor`

```
am_nu_DSI_factor = -1
```

am_nu_SSI_factor ¶

< 0 means use `D_SSI_factor`

```
am_nu_SSI_factor = -1
```

am_nu_SH_factor ¶

< 0 means use `D_SH_factor`

```
am_nu_SH_factor = -1
```

am_nu_ES_factor ¶

< 0 means use `D_ES_factor`

```
am_nu_ES_factor = -1
```

am_nu_GSF_factor ¶

< 0 means use `D_GSF_factor`

```
am_nu_GSF_factor = -1
```

am_nu_ST_factor ¶

< 0 means use `D_ST_factor`

```
am_nu_ST_factor = -1
```

am_nu_visc_factor ¶

< 0 means use `D_visc_factor`. By default = 1 to mix angular momentum.

```
am_nu_visc_factor = 1
```

am_nu_omega_rot_factor ¶

am_nu_omega_non_rot_factor ¶

```
dj/dt = d/dm((4 pi r^2 rho)^2*(am_nu_omega*i_rot*domega/dm + am_nu_j*d
am_nu_omega = am_nu_omega_non_rot_factor*am_nu_non_rot + am_nu_omega_r
```

```
am_nu_omega_rot_factor = 1
am_nu_omega_non_rot_factor = 1
```

am_nu_j_rot_factor ¶

am_nu_j_non_rot_factor ¶

```
dj/dt = d/dm((4 pi r^2 rho)^2*(am_nu_omega*i_rot*domega/dm + am_nu_j*d
am_nu_j = am_nu_j_non_rot_factor*am_nu_non_rot + am_nu_j_rot_factor*am
```

```
am_nu_j_rot_factor = 0
am_nu_j_non_rot_factor = 0
```

set_uniform_am_nu_non_rot ¶

uniform_am_nu_non_rot ¶

You can specify a uniform value for `am_nu_non_rot` by setting this flag true. A large uniform `am_nu` will produce a uniform omega.

```
set_uniform_am_nu_non_rot = .false.
uniform_am_nu_non_rot = 1d20
```

set_min_am_nu_non_rot ¶

min_am_nu_non_rot ¶

You can also specify a minimum `am_nu_non_rot`. `am_nu` will be at least this large.

```
set_min_am_nu_non_rot = .false.
min_am_nu_non_rot = 1d8
```

min_center_Ye_for_min_am_nu_non_rot ¶

`min_am_nu_non_rot` is only used when center Ye >= this.

```
min_center_Ye_for_min_am_nu_non_rot = 0.47d0
```

Each rotationally induced diffusion coefficient has a factor that lets you control it. Value of 1 gives normal strength; value of 0 turns it off.

Note that for regions with brunt N^2 < 0, we set Richardson number to 1, which is > Ri_critical and therefore turns off DSI and SSI.

```
D_DSI_factor = 0
D_SH_factor = 0
D_SSI_factor = 0
D_ES_factor = 0
D_GSF_factor = 0
D_ST_factor = 0
```

D_visc_factor ¶

Kinematic shear viscosity. Should be = 0 because viscosity doesn't mix chemical elements.

```
D_visc_factor = 0
```

am_gradmu_factor ¶

Sensitivity to composition gradients. In calculation of rotational induced mixing, `grad_mu` is multiplied by `am_gradmu_factor`. Value from from Heger et al 2000.

```
am_gradmu_factor = 0.05d0
```

Spatial smoothing is used in calculations of diffusion coefficients. These control the smoothing window widths (number of cells on each side).

```
smooth_D_DSI = 0
smooth_D_SH = 0
```

```
smooth_D_SSI = 0
smooth_D_ES = 0
smooth_D_GSF = 0
smooth_D_ST = 0
smooth_nu_ST = 0
```

time smoothing. Set to 0 to turn off time smoothing.

```
angsmt_D_DSI = 0.0d0
angsmt_D_SH = 0.0d0
angsmt_D_SSI = 0.0d0
angsmt_D_ES = 0.0d0
angsmt_D_GSF = 0.0d0
angsmt_D_ST = 0.2d0
angsmt_nu_ST = 0.2d0
angsml = 1d-3
```

am_time_average ¶

If true, then `D = (D_new + D_old)/2`, where `D_old` is D from previous step and `D_new` is D as calculated for current as if no time smoothing.

```
am_time_average = .false.
```

simple_i_rot_flag ¶

If true, `i_rot = (2/3)*r^2`. If false, use slightly more complex expression that takes into account finite shell thickness. In practice, there doesn't seem to be a significant difference.

```
simple_i_rot_flag = .true.
```

fitted_i_rot_flag ¶

EXPERIMENTAL: If true and simple_i_rot_flag = .false., use fit to i_rot that depends on rotation rate.

```
fitted_i_rot_flag = .false.
```

do_adjust_J_lost ¶

adjust_J_fraction ¶

adjust angular momentum With do_adjust_J_lost = .false., the angular momentum removed via winds from the star corresponds to that contained in the removed layers. However, since j_rot can increase steeply in the very outer layers, very small steps are required to obtain a convergent solution. To avoid this, the do_adjust_J_lost option adjusts the angular momentum content of layers below those removed, such that

```
actual_J_lost = &
   adjust_J_fraction*mass_lost*s% j_rot_avg_surf + &
   (1d0 - adjust_J_fraction)*s% angular_momentum_removed
```

where s% angular_momentum_removed is the angular momentum contained in the removed layers of the star in that step. Note that s% angular_momentum_removed is set to actual_J_lost after this.

The region from which angular momentum is removed is chosen such that at its bottom q<min_q_for_adjust_J_lost, it contains at least min_J_div_delta_J times the angular momentum that needs to be accounted for, and it is at an optical depth below min_tau_for_adjust_J_lost. Angular momentum in these regions is adjusted in such a way that no artificial shear is produced at the inner boundary.

This can also be used to model mass loss mechanisms that remove more angular momentum than mass_lost*s% j_rot_avg_surf, for instance magnetic braking or wind mass loss. In that case, you can use the use_other_j_for_adjust_J_lost option to specify a specific angular momentum of removed material different from j_rot_avg_surf

```
do_adjust_J_lost = .true.
adjust_J_fraction = 1d0
min_q_for_adjust_J_lost = 0.99d0
min_J_div_delta_J = 3d0
min_tau_for_adjust_J_lost = 300d0
```

premix_omega ¶

if premix_omega is true, then do 1/2 of the transport of angular momentum before updating the structure and 1/2 after. otherwise, do all of the transport after updating the structure. RECOMMENDED to turn it on when modelling an accreting star or when using do_adjust_J_lost.

```
premix_omega = .true.
```

recalc_mixing_info_each_substep ¶

if `recalc_mixing_info_each_substep` is true, then recalculate the omega mixing coefficients after each substep of the solve omega mix process.

```
recalc_mixing_info_each_substep = .false.
```

use_fitted_fpft ¶

Use analytical fits to the rotational corrections fp and ft, computed using the Roche potential for a single particle.

```
use_fitted_fpft = .false.
```

w_div_wcrit_min_for_fpft ¶

When use_fitted_fpft = .true., limit fp and ft to their values at this w_div_wcrit

```
w_div_wcrit_min_for_fpft = 0.9999
```

FP_min ¶

FT_min ¶

Lower limits for rotational distortion corrections factors FP and FT. Used for the calculation when use_fitted_fpft = .false., otherwise the limits are set using w_div_wcrit_min_for_fpft

```
FP_min = 0.75d0
FT_min = 0.95d0
```

FP_error_limit ¶

If calculate an fp < this, treat it as an error. Used for the calculation when use_fitted_fpft = .false.

```
FP_error_limit = 0d0
```

FT_error_limit ¶

If calculate an ft < this, treat it as an error. Used for the calculation when use_fitted_fpft = .false.

```
FT_error_limit = 0d0
```

D_mix_rotation_max_logT_full_on ¶

Use rotational components of `D_mix` for locations where logT <= this. For numerical stability, turn off rotational part of `D_mix` at very high T.

```
D_mix_rotation_max_logT_full_on = 9.4d0
```

D_mix_rotation_min_logT_full_off ¶

Drop rotational components of `D_mix` for locations where logT >= this. For numerical stability, turn off rotational part of `D_mix` at very high T.

```
D_mix_rotation_min_logT_full_off = 9.5d0
```

D_omega_max_replacement_fraction ¶

D_omega_growth_rate ¶

D_omega_mixing_rate ¶

D_omega_mixing_across_convection_boundary (previously called D_omega_mixing_in_convection_regions) ¶

```
D_omega_max_replacement_fraction = 0.5d0
D_omega_growth_rate = 1d0
D_omega_mixing_rate = 1d0
D_omega_mixing_across_convection_boundary = .false.
max_q_for_D_omega_zero_in_convection_region = 0.8d0
```

nu_omega_max_replacement_fraction ¶

nu_omega_growth_rate ¶

nu_omega_mixing_rate ¶

nu_omega_mixing_across_convection_boundary ¶

```
nu_omega_max_replacement_fraction = 0.5d0
nu_omega_growth_rate = 1d0
nu_omega_mixing_rate = 1d0
nu_omega_mixing_across_convection_boundary = .false.
max_q_for_nu_omega_zero_in_convection_region = 0.8d0
```

atmosphere boundary conditions ¶

which_atm_option ¶

- `'simple_photosphere'` : don't integrate, just estimate for tau=2/3
- `'Eddington_grey'` : Eddington T-tau integration
- `'Krishna_Swamy'` : Krishna Swamy T-tau integration
- `'solar_Hopf_grey'` : another T(tau), this one tuned to solar data.
- `'tau_100_tables'` : use model atmosphere tables for Pgas and T at tau=100; solar Z only.
- `'tau_10_tables'` : use model atmosphere tables for Pgas and T at tau=10; solar Z only.
- `'tau_1_tables'` : use model atmosphere tables for Pgas and T at tau=1; solar Z only.
- `'tau_1m1_tables'` : use model atmosphere tables for Pgas and T at tau=1e-1; solar Z only.
- `'photosphere_tables'` : use model atmosphere tables for photosphere; range of Z's.
- `'grey_and_kap'` : iterate simple grey to find consistent P, T, and kap at surface
- `'grey_irradiated'` : based on Guillot, T, and Havel, M., A&A 527, A20 (2011).
- `'Paczynski_grey'` : create an atmosphere for given base conditions. inspired by B. Paczynski, 1969, Acta Astr., vol. 19, 1. takes into account dilution when tau < 2/3, and calls mlt to get gradT allowing for convection in atmosphere.
- `'WD_tau_25_tables'` : hydrogen atmosphere tables for cool white dwarfs giving Pgas and T at log10(tau) = 1.4 (tau = 25.11886) Teff goes from 40,000 K down to 2,000K with step of 100 K Log10(g) goes from 9.5 down to 5.5 with step of 0.1. R.D. Rohrmann, L.G. Althaus, and S.O. Kepler, Lyman α wing absorption in cool white dwarf stars, Mon. Not. R. Astron. Soc. 411, 781–791 (2011)
- `'fixed_Teff'` : set Tsurf from Eddington T-tau relation for current surface tau and Teff = `atm_fixed_Teff`. set Psurf = Radiation_Pressure(Tsurf)

- `'fixed_Tsurf'` : get value of Tsurf from control parameter `atm_fixed_Tsurf`. set Teff from Eddington T-tau relation for given Tsurf and tau=2/3 set Psurf = Radiation_Pressure(Tsurf)
- `'fixed_Psurf'` : get value of Psurf from control parameter `atm_fixed_Psurf`. set Tsurf from L and R using `L = 4*pi*R^2*boltz_sigma*T^4`. set Teff using Eddington T-tau relation for tau=2/3 and T=Tsurf.
- `'fixed_Psurf_and_Tsurf'` : get value of Psurf from control parameter `atm_fixed_Psurf`. get value of Tsurf from control parameter `atm_fixed_Tsurf`.

```
which_atm_option = 'simple_photosphere'
```

which_atm_off_table_option ¶

If have selected an atm table as your option, fallback to using this if the args are off the table. `'simple_photosphere'` or `'grey_and_kap'`.

```
which_atm_off_table_option = 'simple_photosphere'
```

atm_fixed_Teff ¶

Set this when using `atm_option = 'fixed_Teff'`

```
atm_fixed_Teff = 0
```

atm_fixed_Tsurf ¶

Set this when using `atm_option = 'fixed_Tsurf'`

```
atm_fixed_Tsurf = 0
```

atm_fixed_Psurf ¶

Set this when using `which_atm_option = 'fixed_Psurf'`

```
atm_fixed_Psurf = -1
```

atm_switch_to_grey_as_backup ¶

If you select a table option, but the args are out of the range of the tables, then this flag determines whether you get an error or the code automatically switches to option = `atm_simple_photosphere` as a backup.

```
atm_switch_to_grey_as_backup = .true.
```

Pextra_factor ¶

Parameter for extra pressure in surface boundary conditions. Pressure at optical depth tau is calculated as `P = tau*g/kap*(1 + Pextra)` Pextra takes into account nonzero radiation pressure at tau=0. The equation for Pextra includes `Pextra_factor`

```
Pextra = Pextra_factor*(kap/tau)*(L/M)/(6d0*pi*clight*cgrav)
```

For certain situations such super eddington L, you may need to increase Pextra to help convergence. e.g. try `Pextra_factor = 2` `Pextra_factor < 0` means use (incorrect) old form `1.6d-4*kap*(L/Lsun)/(M/Msun)`.

```
Pextra_factor = 1
```

atm_grey_and_kap_atol ¶

atm_grey_and_kap_rtol ¶

Relative and absolute tolerance parameters for the `grey_and_kap` option. Iterates on kap until `err = |delta kap|/(atol + rtol*kap) < 1`.

```
atm_grey_and_kap_atol = 1d-7
atm_grey_and_kap_rtol = 1d-7
```

atm_grey_and_kap_max_tries ¶

trace_atm_grey_and_kap ¶

Limit on iterations and trace.

```
atm_grey_and_kap_max_tries = 50
trace_atm_grey_and_kap = .false.
```

`atm_grey_irradiated_atol` and `atm_grey_irradiated_rtol`. Parameters for the `grey_irradiated` option. Absolute and relative error tolerances.

```
atm_grey_irradiated_atol = 1d-4
atm_grey_irradiated_rtol = 1d-4
```

atm_grey_irradiated_T_eq ¶

Equilibrium temperature based on irradiation.

```
irrad_flux = Lstar/(4*pi*orbit**2)
```

- Area of planet in plane perpendicular to `irrad_flux = pi*Rplanet**2`.
- Stellar luminosity received by `planet = irrad_flux*area`.
- This luminosity determines `T_eq`: `T_eq**4 = irrad_flux/(4*sigma)`.

```
atm_grey_irradiated_T_eq = 1000
```

atm_grey_irradiated_kap_v ¶

atm_grey_irradiated_simple_kap_th ¶

Opacity for irradiation. The T(tau) relation for this option depends on the ratio `kap_v/kap_th` where `kap_v` is the planet atmosphere opacity for stellar irradiation, and `kap_th` is the thermal opacity for internally produced radiation. You can either specify the ratio of `kap_v/kap_th`, or you can specify `kap_v` and have the code calc `kap_th` to get the ratio.

```
atm_grey_irradiated_kap_v = 4d-3
atm_grey_irradiated_simple_kap_th = .false.
```

atm_grey_irr_kap_v_div_kap_th ¶

If `atm_grey_irradiated_simple_kap_th` is true, then just set `kap_th = kap_v/kap_v_div_kap_th`. Only used if > 0.

```
atm_grey_irr_kap_v_div_kap_th = 0
```

atm_grey_irradiated_P_surf ¶

Surface pressure ; set to 1 bar in cgs units.

```
atm_grey_irradiated_P_surf = 1d6
```

atm_grey_irradiated_max_tries ¶

Limit on iterations.

```
atm_grey_irradiated_max_tries = 50
```

trace_atm_grey_irradiated ¶

Trace the grey atmosphere.

```
trace_atm_grey_irradiated = .false.
```

atm_int_errtol ¶

dump_int_atm_info_model_number ¶

Parameters for integrate T(tau) and write atm structure at model number to terminal.

```
atm_int_errtol = 1d-7
dump_int_atm_info_model_number = -1111
```

Parameters for `Paczynski_grey`. `create_atm_max_step_size` in units of `log10_tau`.

```
trace_atm_Paczynski_grey = .false.
Paczynski_atm_R_surf_errtol = 3d-4
create_atm_max_step_size = 0.1d0
```

surface_extra_Pgas ¶

Extra gas pressure at surface. Added to surface pressure from atm. In ergs/cm^3.

```
surface_extra_Pgas = 0d0
```

use_atm_PT_at_center_of_surface_cell ¶

The surface boundary conditions for pressure and temperature, compare the model values at the center of the surface cell to values derived from the P and T returned by the atm module. If this flag is true, then the atm values are directly used. If false, then the values from the atm are treated as being for the outer boundary of the surface cell, and those values are used to estimate corresponding values for the cell center for comparison to the model values at the cell center.

Most cases will have this flag false. An example that sets this flag true is a case in which are using a special boundary condition (BC) routine to force a certain entropy for the surface cell. In that situation, it is better to have the special BC directly return P and T for the center of cell 1 to produce the desired entropy.

```
use_atm_PT_at_center_of_surface_cell = .false.
```

use_compression_outer_BC ¶

gradient of compression vanishes at surface

```
see Grott, Chernigovski, Glatzel, 2005.
d_dm(d_dm(r^2*v)) = 0 at surface
by continuity, this is d_dm(d_dt(1/rho)) = 0 at surface
finite volume form is
(1/rho(1) - 1/rho_start(1)) = (1/rho(2) - 1/rho_start(2))
this BC determines the density for surface cell.
```

```
use_compression_outer_BC = .false.
```

use_momentum_outer_BC ¶

use `P_surf` from atm to set pressure gradient at surface in momentum equation calculate v(1) based on pressure difference `P_surf - P(1)`

```
use_momentum_outer_BC = .false.
```

use_zero_Pgas_outer_BC ¶

use `Psurf = Radiation_Pressure(T_start(1))`

```
use_zero_Pgas_outer_BC = .false.
```

use_zero_dLdm_outer_BC ¶

use L(1) = L(2) for T outer BC

```
use_zero_dLdm_outer_BC = .false.
```

use_T_Paczynski_outer_BC ¶

`T_surf^4` is set to `L/(8*pi*boltz_sig*R^2)`

```
use_T_Paczynski_outer_BC = .false.
```

use_T_black_body_outer_BC ¶

`T_surf` is set to `Tsurf_factor*T_black_body(L_surf,R_surf)`

```
use_T_black_body_outer_BC = .false.
```

use_fixed_vsurf_outer_BC ¶

fixed_vsurf ¶

v at outer boundary of model is set to be fixed_vsurf

```
use_fixed_vsurf_outer_BC = .false.
fixed_vsurf = 0
```

use_fixed_L_for_BB_outer_BC ¶

fixed_L_for_BB_outer_BC ¶

for `use_T_black_body_outer_BC` and `use_T_Paczynski_outer_BC`

```
use_fixed_L_for_BB_outer_BC = .false.
fixed_L_for_BB_outer_BC = 0
```

tau_for_L_BB ¶

determines location where get L for use with BB outer BCs use the L at outermost location where tau >= tau_for_L_BB

```
tau_for_L_BB = -1
```

Tsurf_factor ¶

used when `use_momentum_outer_BC` `T_surf` is set to `Tsurf_factor*T_black_body(L_surf,R_surf)`

```
Tsurf_factor = 1
```

irradiation_flux ¶

column_depth_for_irradiation ¶

```
irradiation_flux = 0
column_depth_for_irradiation = -1
```

mass gain or loss ¶

mass_change ¶

Rate of accretion (Msun/year). Negative for mass loss. This only applies when the `wind_scheme = ''`.

```
mass_change = 0d0
```

Enhanced mass loss due to rotation as in Heger, Langer, and Woosley, 2000, ApJ, 528:368-396.

```
  Mdot = Mdot_no_rotation/(1 - Osurf/Osurf_crit)^mdot_omega_power
```

where

```
Osurf = angular velocity at surface
Osurf_crit^2 = (1 - Gamma_edd)*G*M/R^3
```

```
Gamma_edd = kappa*L/(4 pi c G M), Eddington factor
```

Typical value for `mdot_omega_power = 0.43`.

mdot_omega_power ¶

Set to 0 to disable this feature.

```
mdot_omega_power = 0.43d0
```

max_rotational_mdot_boost ¶

This limits the rotational boost.

```
max_rotational_mdot_boost = 1d4
```

max_mdot_jump_for_rotation ¶

Don't increase prev mdot by more that this. NOTE: use `vcrit_max_years_for_timestep` with this.

```
max_mdot_jump_for_rotation = 2
```

lim_trace_rotational_mdot_boost ¶

Output to terminal if boost > this.

```
lim_trace_rotational_mdot_boost = 1d99
```

rotational_mdot_boost_fac ¶

Increase mdot.

```
rotational_mdot_boost_fac = 1d5
```

rotational_mdot_kh_fac ¶

Kelvin-helmholtz boost.

```
rotational_mdot_kh_fac = 0.3d0
```

surf_avg_tau_min ¶

Use mass avg starting from this optical depth.

```
surf_avg_tau_min = 1
```

surf_avg_tau ¶

Use mass avg down to this optical depth.

```
surf_avg_tau = 100
```

hot_wind_scheme ¶

hot_wind_Wolf_Rayet_scheme ¶

cool_wind_RGB_scheme ¶

cool_wind_AGB_scheme ¶

This section replaces the old "`RGB_wind_scheme`" and "`AGB_wind_scheme`" with temperature-dependent hot_wind and cool_wind. You can still use the RGB and AGB wind scheme as before, the functionality remains.

Now you can also select a hot wind scheme that takes effect *above* some temperature, set by `hot_wind_full_on_T`. Similarly, the cool wind scheme has temperature controls that set the temperature *below* which they are relevant (`cool_wind_full_on_T`).

As before, an empty string '' means no wind.

The wind "eta" values, which are constant scaling factors, have all renamed *_wind_eta -> *_scaling_factor.

Here is an example of how to translate an existing inlist from the old style to the new:

```
|        Before                      |             After
|------------------------------------|-----------------------------------
| RGB_wind_scheme = 'Reimers'        |  cool_wind_RGB_scheme = 'Reimers'
| Reimers_wind_eta = 0.1             |  Reimers_scaling_factor = 0.1
| AGB_wind_scheme = 'Blocker'        |  cool_wind_AGB_scheme = 'Blocker'
| Blocker_wind_eta = 0.5             |  Blocker_scaling_factor = 0.5
| RGB_to_AGB_wind_switch = 1d-4      |  RGB_to_AGB_wind_switch = 1d-4
|                                    |
|                                    |  ! only use the cool_wind_scheme
|                                    |  cool_wind_full_on_T = 1d10  !K
|                                    |  hot_wind_full_on_T = 1.1d10 !K
|                                    |  hot_wind_scheme = ''
```

suggested hot and cool wind schemes follow but any valid wind option will work for either hot or cool.

Empty string means no wind

Suggested hot wind options:

- 'Kudritzki'

- ‘Vink’

Suggested cool wind options:

- ‘Reimers’
- ‘Blöcker’
- ‘de Jager’
- ‘van Loon’
- ‘Nieuwenhuijzen’

For now the ‘Dutch’ scheme can be used in either capacity.

```
hot_wind_scheme = ''
cool_wind_RGB_scheme = ''
cool_wind_AGB_scheme = ''
```

cool_wind_full_on_T ¶

hot_wind_full_on_T ¶

use only cool wind schemes for T_phot < `cool_wind_full_on_T` use only hot wind schemes for T_phot > `hot_wind_full_on_T` if `cool_wind_full_on_T` /= `hot_wind_full_on_T` then ramp between these limits requires `hot_wind_full_on_T` > `cool_wind_full_on_T`

```
cool_wind_full_on_T = 0.8d4
hot_wind_full_on_T = 1.2d4
```

RGB_to_AGB_wind_switch ¶

If center hydrogen abundance is < 0.01 and center helium abundance by mass is less than `RGB_to_AGB_wind_switch`, then system will use `AGB_wind_scheme` rather than `RGB_wind_scheme`.

```
RGB_to_AGB_wind_switch = 1d-4
```

The code will automatically choose between an RGB wind and an AGB wind. The following names for the different schemes are recognized:

- ‘Reimers’
- ‘Blocker’
- ‘de Jager’
- ‘van Loon’
- ‘Nieuwenhuijzen’
- ‘Kudritzki’
- ‘Vink’
- ‘Dutch’
- ‘Stern51’
- ‘Grafener’
- ‘other’ — experimental

Reimers_scaling_factor ¶

Reimers mass loss for red giants.

D. Reimers “Problems in Stellar Atmospheres and Envelopes” Baschek, Kegel, Traving (eds), Springer, Berlin, 1975, p. 229.

Parameter for mass loss by Reimers wind prescription. Reimers mdot is `eta*4d-13*L*R/M` (Msun/year), with L, R, and M in solar units. Typical value is 0.5.

```
Reimers_scaling_factor = 0
```

Blocker_scaling_factor = 0 ¶

Blocker’s mass loss for AGB stars.

T. Blocker “Stellar evolution of low and intermediate-mass stars” A&A 297, 727-738 (1995).

Parameter for mass loss by Blocker’s wind prescription. Blocker mdot is `eta*4.83d-9*M**-2.1*L**2.7*4d-13*L*R/M` (Msun/year), with L, R, and M in solar units. Typical value is 0.1d0.

```
Blocker_scaling_factor = 0
```

de_Jager_scaling_factor ¶

de Jager mass loss for various applications. de Jager, C., Nieuwenhuijzen, H., & van der Hucht, K. A. 1988, A&AS, 72, 259. Parameter for mass loss by de Jager wind prescription.

```
de_Jager_scaling_factor = 0d0
```

van_Loon_scaling_factor ¶

see van Loon et al. 2005, A&A, 438, 273 “An empirical formula for the mass-loss rates of dust-enshrouded red supergiants and oxygen-rich Asymptotic Giant Branch stars”

```
van_Loon_scaling_factor = 0d0
```

Kudritzki_scaling_factor ¶

Radiation driven winds of hot stars. See Kudritzki et al, Astron. Astrophys. 219, 205-218 (1989).

```
Kudritzki_scaling_factor = 0d0
```

Nieuwenhuijzen_scaling_factor ¶

See Nieuwenhuijzen, H.; de Jager, C. 1990, A&A, 231, 134.

```
Nieuwenhuijzen_scaling_factor = 0d0
```

Vink_scaling_factor ¶

Vink, J.S., de Koter, A., & Lamers, H.J.G.L.M., 2001, A&A, 369, 574. “Mass-loss predictions for O and B stars as a function of metallicity”

```
Vink_scaling_factor = 0d0
```

Grafener_scaling_factor ¶

Grafener, G. & Hamann, W.-R. 2008, A&A 482, 945 contributed to mesa by Nilou Afsari

```
Grafener_scaling_factor = 0d0
```

Dutch_scaling_factor ¶

The “Dutch” wind scheme for massive stars combines results from several papers, all with authors mostly from the Netherlands.

The particular combination we use is based on Glebbeek, E., et al, A&A 497, 255-264 (2009) [more Dutch authors!]

For Teff > 1e4 and surface H > 0.4 by mass, use Vink et al 2001 Vink, J.S., de Koter, A., & Lamers, H.J.G.L.M., 2001, A&A, 369, 574.

For Teff > 1e4 and surface H < 0.4 by mass, use Nugis & Lamers 2000 Nugis, T.,& Lamers, H.J.G.L.M., 2000, A&A, 360, 227 Some folks use 0.8 for non-rotating mdoels (Maeder & Meynet, 2001).

```
Dutch_scaling_factor = 0d0
```

Dutch_wind_lowT_scheme ¶

For Teff < 1e4

Use de Jager if `Dutch_wind_lowT_scheme = 'de Jager'` de Jager, C., Nieuwenhuijzen, H., & van der Hucht, K. A. 1988, A&AS, 72, 259.

Use van Loon if `Dutch_wind_lowT_scheme = 'van Loon'` van Loon et al. 2005, A&A, 438, 273.

Use Nieuwenhuijzen if `Dutch_wind_lowT_scheme = 'Nieuwenhuijzen'` Nieuwenhuijzen, H.; de Jager, C. 1990, A&A, 231, 134

```
Dutch_wind_lowT_scheme = 'de Jager'
```

Stern51_scaling_factor ¶

wind scheme from Stern

```
Stern51_scaling_factor = 0d0
```

use_accreted_material_j ¶

Angular momentum of accreted material.

```
use_accreted_material_j = .false.
```

If false, then accreted material is given j so that it is rotating at the same angular velocity as the surface. If true, then accreted material is given j = `accreted_material_j`.

```
accreted_material_j = 0
```

no_wind_if_no_rotation ¶

Use this to delay start of wind until after have started rotation.

```
no_wind_if_no_rotation = .false.
```

min_wind ¶

Min wind in Msun/year > 0; ignore this limit if it is <= 0. e.g., might have low level wind even when normal scheme doesn't call for any.

```
min_wind = 0d0
```

max_wind ¶

Max wind in Msun/year > 0; ignore this limit if it is <= 0.

```
max_wind = 0d0
```

For critical rotation mass loss Redo step as needed to find mdot that brings model to just below critical. if `max_mdot_redo_cnt` > 0, and `surf_w_div_w_crit` > `surf_w_div_w_crit_limit`, then recompute the step while increasing mdot, until `surf_w_div_w_crit` < `surf_w_div_w_crit_limit`. Once an upper limit for mdot is found, the solution for mdot is further refined by bisection until it is computed to a tolerance of `surf_w_div_w_crit_tol`. During iterations, mdot is adjusted alternately by multiplication by `mdot_revise_factor`, and by adjusting it by `implicit_mdot_boost*mdot_initial`, where `mdot_initial` is the value of mdot at the first iteration. This is done to deal with mass accreting stars, where mdot might need to change sign for the star to remain below critical.

```
max_mdot_redo_cnt = 0
min_years_dt_for_redo_mdot = 0
surf_w_div_w_crit_limit = 0.99d0
```

```
surf_w_div_w_crit_tol = 0.05d0
mdot_revise_factor = 1.1d0
implicit_mdot_boost = 0.1d0
```

implicit wind computation. ¶

max_tries_for_implicit_wind ¶

The implicit method will modify the mass transfer rate and redo the step until it either finds a solution, or the number of tries hits `max_tries_for_implicit_wind`. If `max_tries_for_implicit_wind = 0`, the wind computation is explicit, meaning that the value of mdot is set using values at the start of the step. This only applies when mdot < 0.

```
max_tries_for_implicit_wind = 0
```

iwind_tolerance ¶

Tolerance for which a solution is considered valid. A solution is valid if

```
abs(explicit_mdot - implicit_mdot) <
       abs(implicit_mdot)*iwind_tolerance
```

where

```
explicit_mdot = mstar_dot at start of step
implicit_mdot = mstar_dot at end of step
```

```
iwind_tolerance = 1d-3
```

iwind_lambda ¶

If do not satisfy tolerance, redo with a different mdot as follows:

```
    mstar_dot = explicit_mdot + &
       iwind_lambda*(implicit_mdot - explicit_mdot)
```

```
iwind_lambda = 1d0
```

remove H wind ¶

remove_H_wind_mdot ¶

This wind removes surface material until reaching a target total H mass for the star. Max rate of removal in Msun/year ; only applies if this is > 0.

```
remove_H_wind_mdot = 0d0
```

remove_H_wind_H_mass_limit ¶

This wind removes surface material until reaching a target total H mass for the star. Turn off this wind when total H mass < this limit (Msun units).

```
remove_H_wind_H_mass_limit = 0
```

super_eddington_scaling_factor ¶

For super eddington wind we use Ledd averaged by mass to optical depth tau = `surf_avg_tau`.

```
super_eddington_scaling_factor = 0
```

super_eddington_wind_Ledd_factor ¶

Parameter for mass loss driven by super Eddington luminosity. Divide L by this factor when computing super Eddington wind, e.g., if this is 2, then only get wind when L/2 > Ledd.

```
super_eddington_wind_Ledd_factor = 1
```

wind_boost_full_off_L_div_Ledd ¶

Boost off for L/Ledd <= this (set large to disable this). This alternative form is used when `super_eddington_scaling_factor` == 0.

```
wind_boost_full_off_L_div_Ledd = 1.5d0
```

wind_boost_full_on_L_div_Ledd ¶

Do max boost for L/Ledd >= this. This alternative form is used when `super_eddington_scaling_factor` == 0.

```
wind_boost_full_on_L_div_Ledd = 5
```

super_eddington_wind_max_boost ¶

Multiply wind mdot by up to this amount. This alternative form is used when `super_eddington_scaling_factor` == 0.

```
super_eddington_wind_max_boost = 1
```

trace_super_eddington_wind_boost ¶

Send super eddington wind information to terminal.

```
trace_super_eddington_wind_boost = .false.
```

mass_change_full_on_dt ¶

mass_change_full_off_dt ¶

These params provide the option to turn off mass change when have very small timesteps. Between `mass_change_full_on_dt` and `mass_change_full_off_dt` mass change is gradually reduced. Units in seconds.

```
mass_change_full_on_dt = 1d-99
mass_change_full_off_dt = 1d-99
```

trace_dt_control_mass_change ¶

```
trace_dt_control_mass_change = .false.
```

min_abs_mdot_for_change_limits ¶

Only apply limits if abs(prev mdot) > this limit. These limit the change in mdot from one step to the next.

```
min_abs_mdot_for_change_limits = 1d-14
```

max_abs_mdot_factor ¶

Only allow abs(mdot) to increase by this factor per timestep.

```
max_abs_mdot_factor = 2
```

min_abs_mdot_factor ¶

Only allow abs(mdot) to decrease by this factor per timestep.

```
min_abs_mdot_factor = 0.5d0
```

max_star_mass_for_gain ¶

Automatic stops for mass loss/gain in Msun units (negative means ignore this parameter). Turn off mass gain when star mass reaches this limit.

```
max_star_mass_for_gain = -1
```

min_star_mass_for_loss ¶

Automatic stops for mass loss/gain in Msun units (negative means ignore this parameter). Turn off mass loss when star mass reaches this limit.

```
min_star_mass_for_loss = -1
```

max_T_center_for_any_mass_loss ¶

No mass loss for T center > this.

```
max_T_center_for_any_mass_loss = 2d9
```

max_T_center_for_full_mass_loss ¶

No reduction in mass loss for T center <= this. This must be <= `max_T_center_for_full_mass_loss`. Reduce mass loss rate to 0 as T center climbs from `max_for_full` to `max_for_any`. The idea behind this is that during final stages of burning, there is so little time left in the life of the star, that any mass loss to winds will be negligible, but the inclusion of that insignificant mass loss can actually make convergence more difficult, so you are better off without it.

```
max_T_center_for_full_mass_loss = 1d9
```

wind_envelope_limit ¶

Winds automatically shut off when the hydrogen rich envelope mass is less than this limit. The value of `h1_boundary_limit` defines what is considered to be hydrogen poor. Mass in Msun units.

```
wind_envelope_limit = -1
```

rlo_scaling_factor ¶

Amplitude of mass loss. “rlo” wind scheme provides a simple radius-determined-wind with exponential increase.

```
rlo_scaling_factor = 0
```

rlo_wind_min_L ¶

Only on when L > this limit. (Lsun)

```
rlo_wind_min_L = 1d-6
```

rlo_wind_max_Teff ¶

Only on when Teff < this limit.

```
rlo_wind_max_Teff = 1d99
```

rlo_wind_roche_lobe_radius ¶

Only on when R > this (Rsun).

```
rlo_wind_roche_lobe_radius = 0.40d0
```

rlo_wind_base_mdot ¶

Base rate of mass loss when R = roche lobe radius (Msun/year).

```
rlo_wind_base_mdot = 1d-3
```

rlo_wind_scale_height ¶

Determines exponential growth rate of mass loss (Rsun).

```
rlo_wind_scale_height = 1d-1
```

roche_lobe_xfer_full_on ¶

Full accretion when R/RL <= this. Limit accretion when Roche lobe is nearing full (only with `rlo_scaling_factor` > 0).

```
roche_lobe_xfer_full_on = 0.5d0
```

roche_lobe_xfer_full_off ¶

No accretion when R/RL >= this.

```
roche_lobe_xfer_full_off = 1.0d0
```

nova_scaling_factor ¶

Amplitude of wind. “nova” wind is scheme used in Kato and Hachisu, ApJ 437:802-826, 1994. (eqn 23). This only applies when `nova_scaling_factor` > 0.

```
nova_scaling_factor = 0
```

nova_wind_b ¶

Wind parameter

```
nova_wind_b = 0
```

nova_wind_max_Teff ¶

Only on when Teff < this limit.

```
nova_wind_max_Teff = 0
```

nova_wind_min_L ¶

only on when L > this limit. (Lsun)

```
nova_wind_min_L = 0
```

nova_min_Teff_for_accretion ¶

When `nova_scaling_factor /= 0` and `Teff < this` and `L > nova_wind_min_L`, no accretion.

```
nova_min_Teff_for_accretion = 0
```

nova_roche_lobe_radius ¶

units in Rsun

```
nova_roche_lobe_radius = 0
```

nova_RLO_mdot ¶

roche lobe overflow mdot, Msun/year

```
nova_RLO_mdot = 0
```

flash_wind_mdot ¶

Rate of mass ejection in Msun/year. “flash” wind is scheme used in Kato, Saio, and Hachisu, ApJ 340:509-517, 1989. This only applies when `flash_wind_mdot` > 0.

```
flash_wind_mdot = -1
```

flash_wind_starts ¶

Wind starts when R >= this limit (Rsun units).

```
flash_wind_starts = -1
```

flash_wind_declines ¶

Wind starts to decline when R <= this limit (Rsun units).

```
flash_wind_declines = -1
```

flash_wind_full_off ¶

Wind full off when R <= this limit (Rsun units).

```
flash_wind_full_off = -1
```

controls for adjust_mass ¶

max_logT_for_k_below_const_q ¶

max_q_for_k_below_const_q ¶

min_q_for_k_below_const_q ¶

Move `k_below_const_q` inward from surface until `q(k) <= max_q`. Then continue moving inward until reach `logT(k) >= max_logT` or `q(k) <= min_q`.

```
max_logT_for_k_below_const_q = 5
max_q_for_k_below_const_q = 1.0d0
min_q_for_k_below_const_q = 0.999d0
```

max_logT_for_k_const_mass ¶

max_q_for_k_const_mass ¶

min_q_for_k_const_mass ¶

Move `k_below_const_q` inward from `k_below_const_q+1` until `q(k) <= max_q`. Then continue moving inward until reach `logT(k) >= max_logT` or `q(k) <= min_q`.

```
max_logT_for_k_const_mass = 6
max_q_for_k_const_mass = 1.0d0
min_q_for_k_const_mass = 0.995d0
```

composition controls ¶

accrete_same_as_surface ¶

If true, composition of accreted material is identical to the current surface composition.

```
accrete_same_as_surface = .true.
```

accrete_given_mass_fractions ¶

If true, use the following mass fractions – they must add to 1.0.

```
accrete_given_mass_fractions = .false.
```

num_accretion_species ¶

Up to `max_num_accretion_species`.

```
num_accretion_species = 0
```

accretion_species_id ¶

Isotope name as defined in `chem_def`.

```
accretion_species_id(1) = ''
```

accretion_species_xa ¶

mass fraction

```
accretion_species_xa(1) = 0
```

otherwise, use the following composition

accretion_h1 ¶

Hydrogen mass fraction.

```
accretion_h1 = 0
```

accretion_h2 ¶

If no h2 in current net, then this is automatically added to h1.

```
accretion_h2 = 0
```

accretion_he3 ¶

he3 mass fraction

```
accretion_he3 = 0
```

accretion_he4 ¶

he4 mass fraction

```
accretion_he4 = 0
```

accretion_zfracs = ¶

One of the following identifiers for different Z fractions from `chem_def`.

- `AG89_zfracs = 1`, Anders & Grevesse 1989
- `GN93_zfracs = 2`, Grevesse & Noels 1993
- `GS98_zfracs = 3`, Grevesse & Sauval 1998
- `L03_zfracs = 4`, Lodders 2003
- `AGS05_zfracs = 5`, Asplund, Grevesse & Sauval 2005

or set `accretion_zfracs = 0` to use the following list of z fractions

```
accretion_zfracs = -1
```

accretion_dump_missing_metals_into_heaviest ¶

this controls the treatment metals that are not included in the current net. if this flag is true, then the mass fractions of missing metals are added to the mass fraction of the most massive metal included in the net. if this flag is false, then the mass fractions of the metals in the net are renormalized to make up for the total mass fraction of missing metals.

```
accretion_dump_missing_metals_into_heaviest = .true.
```

Special list of z fractions. If you use these, they must add to 1.0.

```
z_fraction_li = 0
z_fraction_be = 0
```

```
z_fraction_b = 0
z_fraction_c = 0
z_fraction_n = 0
z_fraction_o = 0
z_fraction_f = 0
z_fraction_ne = 0
z_fraction_na = 0
z_fraction_mg = 0
z_fraction_al = 0
z_fraction_si = 0
z_fraction_p = 0
z_fraction_s = 0
z_fraction_cl = 0
z_fraction_ar = 0
z_fraction_k = 0
z_fraction_ca = 0
z_fraction_sc = 0
z_fraction_ti = 0
z_fraction_v = 0
z_fraction_cr = 0
z_fraction_mn = 0
z_fraction_fe = 0
z_fraction_co = 0
z_fraction_ni = 0
z_fraction_cu = 0
z_fraction_zn = 0
```

lgT_lo_for_set_new_abundances ¶

lgT_hi_for_set_new_abundances ¶

Composition controls for `set_new_abundances`.

```
lgT_lo_for_set_new_abundances = 5.2d0
lgT_hi_for_set_new_abundances = 5.5d0
```

pure_fe56_limit ¶

Pure fe56 for base of ns envelope. If mass fraction of fe56 > this, convert cell to pure fe56.

```
pure_fe56_limit = 0.999999d0
```

mesh adjustment ¶

max_allowed_nz ¶

Maximum number of grid points allowed.

```
max_allowed_nz = 8000
```

remesh_max_allowed_logT ¶

Turn off remesh if any cell has logT > this.

```
remesh_max_allowed_logT = 1d99
```

mesh_max_allowed_ratio ¶

Must be >= 2.5. Max ratio for mass of adjacent cells. If have ratio exceeding this, split the larger cell.

```
mesh_max_allowed_ratio = 2.5d0
```

max_delta_x_for_merge ¶

Don’t merge neighboring cells if any abundance differs by more than this.

```
max_delta_x_for_merge = 0.1d0
```

mesh_delta_coeff ¶

A larger value increases the max allowed deltas and decreases the number of grid points. and a smaller does the opposite. E.g., you’ll roughly double the number of grid points if you cut `mesh_delta_coeff` in half. Don’t expect it to exacly double the number however since other parameters in addition to gradients also influence the details of the grid spacing.

```
mesh_delta_coeff = 1.0d0
```

mesh_delta_coeff_for_highT ¶

Use different `mesh_delta_coeff` at higher temperatures.

```
mesh_delta_coeff_for_highT = 3.0d0
```

logT_max_for_standard_mesh_delta_coeff ¶

Use `mesh_delta_coeff` for center logT <= this. This value should be less than `logT_min_for_highT_mesh_delta_coeff`.

```
logT_max_for_standard_mesh_delta_coeff = 9.0d0
```

logT_min_for_highT_mesh_delta_coeff ¶

Use `mesh_delta_coeff_for_highT` for center logT >= this. Linearly interpolate in logT for intermediate center temperatures.

```
logT_min_for_highT_mesh_delta_coeff = 9.5d0
```

mesh_Pgas_div_P_exponent ¶

Multiply `mesh_delta_coeff` by (Pgas/Ptotal) to this power.

```
mesh_Pgas_div_P_exponent = 0
```

mesh_delta_coeff_pre_ms ¶

Multiply `mesh_delta_coeff` by this when center XH > 0.5 and `lg_LH < lg_L - 1`.

```
mesh_delta_coeff_pre_ms = 1
```

max_dq ¶

Max size for cell as fraction of total mass.

```
max_dq = 1d-2
```

min_dq ¶

Min size for cell as fraction of total mass.

```
min_dq = 1d-14
```

min_dq_for_xa ¶

Min size for splitting because of composition gradient.

```
min_dq_for_xa = 1d-14
```

mesh_min_dlnR ¶

Limit on difference in lnR across cell for mesh refinement. Do not make this smaller than about 1d-14 or will fail with numerical problems.

```
mesh_min_dlnR = 1d-9
```

merge_if_dlnR_too_small ¶

If true, mesh adjustment will force merge if difference in lnR across cell is too small.

```
merge_if_dlnR_too_small = .false.
```

mesh_min_dr_div_dRstar ¶

Limit on relative radial extent for mesh refinement. dRstar = s% r(1) - s% R_center Don't split if dr/dRstar would drop below this limit.

```
mesh_min_dr_div_dRstar = -1
```

merge_if_dr_div_dRstar_too_small ¶

If true, mesh adjustment will force merge if `dr_div_dRstar` too small.

```
merge_if_dr_div_dRstar_too_small = .true.
```

mesh_min_dr_div_cs ¶

Limit (in seconds) on sound crossing time for mesh refinement. Don't split if sound crossing time would drop below this limit.

```
mesh_min_dr_div_cs = -1
```

merge_if_dr_div_cs_too_small ¶

If true, mesh adjustment will force merge if `dr_div_cs` too small.

```
merge_if_dr_div_cs_too_small = .true.
```

max_center_cell_dq ¶

Largest allowed dq at center.

```
max_center_cell_dq = 1d-7
```

max_surface_cell_dq ¶

Largest allowed dq at surface.

```
max_surface_cell_dq = 1d-12
```

max_num_subcells ¶

Limits number of new cells from 1 old one.

```
max_num_subcells = 2
```

max_num_merge_cells ¶

Limits number of old cells to merge into 1 new one.

```
max_num_merge_cells = 2
```

mesh_adjust_use_quadratic ¶

Linear or quadratic reconstruction polynomials for mesh adjustments.

```
mesh_adjust_use_quadratic = .true.
```

mesh_adjust_get_T_from_E ¶

If true, then use internal energy conservation to set new temperature. If false, just use average temperature based on reconstruction polynomials.

```
mesh_adjust_get_T_from_E = .true.
```

P_function_weight ¶

Pressure gradient, `P_function = P_function_weight*log10(P)`.

```
P_function_weight = 40
```

T_function1_weight ¶

Temperature gradient, `T_function1 = T_function1_weight*log10(T)`. NOTE: The T gradient mesh controls below seems to be necessary to allow burning that starts off center to be able to reach the center. You can see this in the `pre_zahb test_suite` case if you try running it without the T function. The center temperature will fail to rise.

```
T_function1_weight = 110
```

T_function2_weight ¶

T_function2_param ¶

```
T_function2 = T_function2_weight*log10(T / (T + T_function2_param))
```

Largest change in `T_function2` happens around `T = T_function2_param`. Default value puts this in the envelope ionization region.

```
T_function2_weight = 0
T_function2_param = 2d4
```

R_function_weight ¶

R_function_param ¶

log radius gradient

```
R_function = R_function_weight*log10(1 + (r/Rsun)/R_function_param)
```

```
R_function_weight = 0
R_function_param = 1d-4
```

R_function2_weight ¶

R_function2_param1 ¶

R_function2_param2 ¶

```
R_function2 = R_function2_weight*min(R_function2_param1,max(R_function
```

where Rstar = radius of outer edge of model.

```
R_function2_weight = 0
R_function2_param1 = 0.4d0
R_function2_param2 = 0
```

R_function3_weight ¶

radius gradient

```
R_function3 = R_function3_weight*(r/Rstar)
```

```
R_function3_weight = 0
```

M_function_weight ¶

M_function_param ¶

log mass gradient

```
M_function = M_function_weight*log10(1 + (m/Msun)/M_function_param)
```

```
M_function_weight = 0
M_function_param = 1d-6
```

gradT_function_weight ¶

gradT gradient, `gradT_function = gradT_function_weight*gradT`

```
gradT_function_weight = 0
```

log_tau_function_weight ¶

log_tau gradient (optical depth)

```
log_tau_function = log_tau_function_weight*log10(tau)
```

```
log_tau_function_weight = 0
```

log_kap_function_weight ¶

log_kap gradient (optical depth)

```
log_kap_function = log_kap_function_weight*log10(kap)
```

```
log_kap_function_weight = 0
```

omega_function_weight ¶

omega gradient (rotation omega in rad/sec)

```
omega_function = omega_function_weight*log10(omega)
```

```
omega_function_weight = 0
```

gam_function_weight ¶

gam_function_param1 ¶

gam_function_param2 ¶

For extra resolution around liquid/solid transition.

```
gam = plasma interaction parameter
gam_function = gam_function_weight*tanh((gam - gam_function_param1)/ga
```

```
gam_function_weight = 0
gam_function_param1 = 170
gam_function_param2 = 20
```

xa_function_species ¶

xa_function_weight ¶

Mass fraction gradients.

```
xa_function = xa_function_weight*log10(xa + xa_function_param),
```

Up to `num_xa_function` of these - see `star_def` for value of `num_xa_function`. 0 length string means skip, otherwise name of nuclide as defined in `chem_def`. weight <= 0 means skip.

```
xa_function_species(:) = ''
xa_function_weight(:) = 0
```

```
xa_function_species(1) = 'he4'
xa_function_weight(1) = 30
xa_function_param(1) = 1d-2
```

xa_mesh_delta_coeff ¶

Useful if you want to increase `mesh_delta_coeff` during advanced burning. If `xa_function_species(j)` has the largest atomic number in current set of species, then multiply `mesh_delta_coeff` by `xa_mesh_delta_coeff(j)`.

```
xa_mesh_delta_coeff(:) = 1
```

“Indirect” mesh controls work by increasing sensitivity in selected regions. They work in the same way as `mesh_delta_coeff` – values less than 1.0 mean smaller allowed jumps in mesh functions and hence smaller grid points and higher resolution. But whereas `mesh_delta_coeff` applies uniformly to all cells, the “extra” coefficients can vary in value from one cell to the next.

xtra_coef_above_xtrans ¶

xtra_coef_below_xtrans ¶

Multiply `mesh_delta_coeff` near any change in most abundant species by this factor. Value < 1 gives increased resolution.

```
xtra_coef_above_xtrans = 1
xtra_coef_below_xtrans = 1
```

xtra_dist_above_xtrans ¶

xtra_dist_below_xtrans ¶

Increase resolution up to this distance away from the abundance transition with distance measured in units of the pressure scale height at the boundary.

```
xtra_dist_above_xtrans = 0.2d0
xtra_dist_below_xtrans = 0.2d0
```

mesh_logX_species ¶

mesh_logX_min_for_extra ¶

Increase resolution at points with large abs(dlogX/dlogP); logX = log10(X mass fraction).

```
mesh_logX_species(1) = ''
mesh_logX_min_for_extra(1) = -6
```

mesh_dlogX_dlogP_extra(1) ¶

mesh_dlogX_dlogP_full_on(1) ¶

mesh_dlogX_dlogP_full_off(1) ¶

Only increase resolution if `logX >= mesh_logX_min_for_extra`. Make `mesh_dlogX_dlogP_extra < 1` for smaller allowed change in logP and hence higher resolution. Full effect if `abs(dlogX/dlogP) >= mesh_dlogX_dlogP_full_on`. No effect if `abs(dlogX/dlogP)) <= mesh_dlogX_dlogP_full_off`. Up to `num_mesh_logX` of these (see `star_def` for value of `num_mesh_logX`).

```
mesh_dlogX_dlogP_extra(1) = 1
mesh_dlogX_dlogP_full_on(1) = 2
mesh_dlogX_dlogP_full_off(1) = 1
```

Multiply `mesh_delta_coeff` near convection zone boundary (czb) by the following factors. Value < 1 gives increased resolution.

xtra_coef_czb_full_on ¶

xtra_coef_czb_full_off ¶

The center mass fraction of he4 is used to control this extra coefficient. The default settings limit the application to after center he4 is depleted.

- if center he4 < `xtra_coef_czb_full_on`, then use xtra coef's
- if center he4 > `xtra_coef_czb_full_off`, then don't use xtra coef's

a more verbose form of these names might be the following:

```
xtra_coef_czb_full_on_if_center_he4_below_this
xtra_coef_czb_full_off_if_center_he4_above_this
```

```
xtra_coef_czb_full_on = 1d-4
xtra_coef_czb_full_off = 0.1d0
```

xtra_coef_{above | below}_{lower | upper}_{nonburn | hburn | heburn | zburn}_czb ¶

Make these < 1 to increase resolution.

xtra_dist_{above | below}_{lower | upper}_{nonburn | hburn | heburn | zburn}_czb ¶

Increase resolution up to this distance away from the convective zone boundary, with distance measured in units of the pressure scale height at the boundary.

```
xtra_coef_a_l_nb_czb = 1
xtra_dist_a_l_nb_czb = 0.2d0
xtra_coef_b_l_nb_czb = 1
xtra_dist_b_l_nb_czb = 0.2d0
```

```
xtra_coef_a_u_nb_czb = 1
xtra_dist_a_u_nb_czb = 0.2d0
xtra_coef_b_u_nb_czb = 1
xtra_dist_b_u_nb_czb = 0.2d0
```

```
xtra_coef_a_l_hb_czb = 1
xtra_dist_a_l_hb_czb = 0.2d0
xtra_coef_b_l_hb_czb = 1
xtra_dist_b_l_hb_czb = 0.2d0
```

```
xtra_coef_a_u_hb_czb = 1
xtra_dist_a_u_hb_czb = 0.2d0
xtra_coef_b_u_hb_czb = 1
xtra_dist_b_u_hb_czb = 0.2d0
```

```
xtra_coef_a_l_heb_czb = 1
xtra_dist_a_l_heb_czb = 0.2d0
xtra_coef_b_l_heb_czb = 1
xtra_dist_b_l_heb_czb = 0.2d0
```

```
xtra_coef_a_u_heb_czb = 1
xtra_dist_a_u_heb_czb = 0.2d0
xtra_coef_b_u_heb_czb = 1
xtra_dist_b_u_heb_czb = 0.2d0
```

```
xtra_coef_a_l_zb_czb = 1
xtra_dist_a_l_zb_czb = 0.2d0
xtra_coef_b_l_zb_czb = 1
xtra_dist_b_l_zb_czb = 0.2d0
```

```
xtra_coef_a_u_zb_czb = 1
xtra_dist_a_u_zb_czb = 0.2d0
xtra_coef_b_u_zb_czb = 1
xtra_dist_b_u_zb_czb = 0.2d0
```

xtra_coef_scz_above_{nonburn | hburn | heburn | zburn}_cz ¶

Make these < 1 to increase resolution in semiconvective region adjacent to convective region. e.g., `xtra_coef_scz_above_nb_cz` is extra coef for semiconvectize zone above a non-burn convective zone.

```
xtra_coef_scz_above_nb_cz = 1
xtra_coef_scz_above_hb_cz = 1
xtra_coef_scz_above_heb_cz = 1
xtra_coef_scz_above_zb_cz = 1
```

Multiply `mesh_delta_coeff` in overshooting regions by the following factors. Value < 1 gives increased resolution.

xtra_coef_os_full_on ¶

xtra_coef_os_full_off ¶

The center mass fraction of he4 is used to control this extra coefficient. The default settings limit the application to after center he4 is depleted.

- if center he4 < `xtra_coef_os_full_on`, then use `xtra_coef` coef's

- if center he4 > `xtra_coef_os_full_off`, then don't use `xtra_coef` coef's

```
xtra_coef_os_full_on = 1d-4
xtra_coef_os_full_off = 0.1d0
```

xtra_coef_os_{above | below}_{nonburn | hburn | heburn | zburn} ¶

Make these < 1 to increase resolution.

```
xtra_coef_os_above_nonburn = 1
xtra_coef_os_below_nonburn = 1
xtra_coef_os_above_burn_h = 1
xtra_coef_os_below_burn_h = 1
xtra_coef_os_above_burn_he = 1
xtra_coef_os_below_burn_he = 1
xtra_coef_os_above_burn_z = 1
xtra_coef_os_below_burn_z = 1
```

xtra_dist_os_{above | below}_{nonburn | hburn | heburn | zburn} ¶

Continue to increase resolution for this distance beyond the edge of the overshooting region, with distance measured in units of the pressure scale height at the edge of the overshooting region. This applies to both edges of the overshooting region.

```
xtra_dist_os_above_nonburn = 0.2d0
xtra_dist_os_below_nonburn = 0.2d0
xtra_dist_os_above_burn_h = 0.2d0
xtra_dist_os_below_burn_h = 0.2d0
xtra_dist_os_above_burn_he = 0.2d0
xtra_dist_os_below_burn_he = 0.2d0
xtra_dist_os_above_burn_z = 0.2d0
xtra_dist_os_below_burn_z = 0.2d0
```

Increase resolution at points with large `abs(dlog_eps/dlogP)` for nuclear power eps (ergs/g/sec). At any particular location, only use eps nuc category with max local value e.g., only use `mesh_dlog_pp_dlogP_extra` at points where pp is the max burn source.

mesh_dlog_eps_min_for_extra ¶

Only increase resolution if `log_eps >= mesh_dlog_eps_min_for_extra`.

```
mesh_dlog_eps_min_for_extra = -2
```

mesh_dlog_eps_dlogP_full_on ¶

Full effect if `abs(dlog_eps/dlogP) >= mesh_dlog_eps_dlogP_full_on`.

```
mesh_dlog_eps_dlogP_full_on = 4
```

mesh_dlog_eps_dlogP_full_off ¶

No effect if `abs(dlog_eps/dlogP)) <= mesh_dlog_eps_dlogP_full_off`.

```
mesh_dlog_eps_dlogP_full_off = 1
```

Multiply the allowed change between adjacent cells by the following factors; (small factor => smaller allowed change => more cells).

pp and cno burning

```
mesh_dlog_pp_dlogP_extra = 0.25d0
mesh_dlog_cno_dlogP_extra = 0.25d0
```

triple alpha, c, n, and o burning

```
mesh_dlog_3alf_dlogP_extra = 0.25d0
mesh_dlog_burn_c_dlogP_extra = 0.25d0
mesh_dlog_burn_n_dlogP_extra = 0.25d0
mesh_dlog_burn_o_dlogP_extra = 0.25d0
```

ne, na, and mg burning

```
mesh_dlog_burn_ne_dlogP_extra = 0.25d0
mesh_dlog_burn_na_dlogP_extra = 0.25d0
mesh_dlog_burn_mg_dlogP_extra = 0.25d0
```

c12+c12. c12+o16, and o16+o16 burning

```
mesh_dlog_cc_dlogP_extra = 0.25d0
mesh_dlog_co_dlogP_extra = 0.25d0
mesh_dlog_oo_dlogP_extra = 0.25d0
```

si to iron alog alpha chain burning

```
mesh_dlog_burn_si_dlogP_extra = 0.25d0
mesh_dlog_burn_s_dlogP_extra = 0.25d0
mesh_dlog_burn_ar_dlogP_extra = 0.25d0
mesh_dlog_burn_ca_dlogP_extra = 0.25d0
mesh_dlog_burn_ti_dlogP_extra = 0.25d0
mesh_dlog_burn_cr_dlogP_extra = 0.25d0
mesh_dlog_burn_fe_dlogP_extra = 0.25d0
```

photodisintegration burning

```
mesh_dlog_pnhe4_dlogP_extra = 0.25d0
mesh_dlog_other_dlogP_extra = 0.25d0
mesh_dlog_photo_dlogP_extra = 1
```

convective_bdy_weight ¶

convective_bdy_dq_limit ¶

convective_bdy_min_dt_yrs ¶

Mesh function to enhance resolution near convective boundaries including regions that are newly nonconvective because of moving boundary. EXPERIMENTAL

```
convective_bdy_weight = 0
convective_bdy_dq_limit = 1d-4
convective_bdy_min_dt_yrs = 1d-3
```

trace_mesh_adjust_error_in_conservation ¶

If true, report relative errors for total PE, KE, and IE. (potential, kinetic, internal).

```
trace_mesh_adjust_error_in_conservation = .false.
```

okay_to_remesh ¶

If false, then no remeshing.

```
okay_to_remesh = .true.
```

remesh_dt_limit ¶

No remesh if `dt < remesh_dt_limit`, in seconds.

```
remesh_dt_limit = -1
```

remesh_log_L_nuc_burn_min ¶

No mesh adjustments when `log10(L_nuc_burn_total)` is less than this. By default, this turns off mesh changes during the early pre-MS.

```
remesh_log_L_nuc_burn_min = -50
```

use_split_merge_amr ¶

```
use_split_merge_amr = .false.
```

use_split_merge_amr_log_zoning ¶

if true, target is even grid spacing in logr if false, target is even grid spacing in r

```
split_merge_amr_log_zoning = .true.
```

split_merge_amr_nz_baseline ¶

```
split_merge_amr_nz_baseline = 1000
```

split_merge_amr_MaxLong ¶

split cell if ratio of actual/desired dr is > this; ignore if <= 0

```
split_merge_amr_MaxLong = 1.5d0
```

split_merge_amr_MaxShort ¶

merge cell if ratio of desired/actual dr is > this; ignore if <= 0

```
split_merge_amr_MaxShort = 4d0
```

merge_amr_max_abs_du_div_cs ¶

```
merge_amr_max_abs_du_div_cs = 0.1d0
```

merge_amr_inhibit_at_jumps ¶

```
merge_amr_inhibit_at_jumps = .false.
```

split_merge_amr_dq_min ¶

do not split if dq for cell is < this

```
split_merge_amr_dq_min = 1d-14
```

split_merge_amr_max_iters ¶

```
split_merge_amr_max_iters = 100
```

split_merge_amr_okay_to_split_1 ¶

split_merge_amr_okay_to_split_nz ¶

```
split_merge_amr_okay_to_split_1 = .true.
split_merge_amr_okay_to_split_nz = .true.
```

equal_split_density_amr ¶

```
equal_split_density_amr = .false.
```

trace_split_merge_amr ¶

```
trace_split_merge_amr = .false.
```

nuclear reaction controls ¶

default_net_name ¶

Name of base reaction network. Each net corresponds to a file in `$MESA_DIR/data/net_data/nets`. Look in that directory to see your network options, or learn how to create your own net.

```
default_net_name = 'basic.net'
```

screening_mode ¶

- empty string means no screening
- `'classic'` : DeWitt, Graboske, Cooper, “Screening Factors for Nuclear Reactions. I. General Theory”, ApJ, 181:439-456, 1973. Graboske, DeWitt, Grossman, Cooper, “Screening Factors for Nuclear Reactions. II. Intermediate Screening and Astrophysical Applications”, ApJ, 181:457-474, 1973.
- `' extended'` : extends the Graboske method using results from Alastuey and Jancovici (1978), along with plasma parameters from Itoh et al (1979) for strong screening.
- `'salpeter'` : weak screening only. following Salpeter (1954), with equations (4-215) and (4-221) of Clayton (1968).

```
screening_mode = 'extended'
```

net_logTcut_lo ¶

strong rates are zero `logT < logTcut_lo` use default from net if this is <= 0

```
net_logTcut_lo = -1
```

net_logTcut_lim ¶

strong rates cutoff smoothly for `logT < logTcut_lim` use default from net if this is <= 0

```
net_logTcut_lim = -1
```

max_abar_for_burning ¶

if abar > this, suppress all burning e.g., if want an “inert” core heavy elements, set this to 55 or, if want to turn off the net, set this to -1

```
max_abar_for_burning = 199
```

dxdt_nuc_factor ¶

Control for abundance changes by burning. Changes `dxdt_nuc` (rate of change of abundances) without changing the rates or `eps_nuc` (rate of energy generation).

```
dxdt_nuc_factor = 1
```

weak_rate_factor ¶

all weak rates are multiplied by this factor

```
weak_rate_factor = 1
```

reaction_neuQs_factor ¶

all neutrino Q factors are multiplied by this factor

```
reaction_neuQs_factor = 1
```

nonlocal_NiCo_kap_gamma ¶

```
nonlocal_NiCo_kap_gamma = 0
```

nonlocal_NiCo_decay_heat ¶

if true, do non-local deposition of gamma-ray energy from Ni56 and Co56 decays. only for approx nets including co56. intended for use with stripped envelope supernovae.

```
nonlocal_NiCo_decay_heat = .false.
```

dtau_gamma_NiCo_decay_heat ¶

```
dtau_gamma_NiCo_decay_heat = 1d0
```

max_logT_for_net ¶

```
max_logT_for_net = 10.2d0
```

element diffusion ¶

gravitational settling and chemical diffusion.

show_diffusion_info ¶

terminal output for diffusion

```
show_diffusion_info = .false.
```

show_diffusion_substep_info ¶

terminal output for diffusion

```
show_diffusion_substep_info = .false.
```

show_diffusion_timing ¶

show time for each call on diffusion

```
show_diffusion_timing = .false.
```

do_element_diffusion ¶

determines whether or not we do element diffusion

```
do_element_diffusion = .false.
```

diffusion_dt_limit ¶

no element diffusion if dt < this limit (in seconds)

```
diffusion_dt_limit = 3.15d7
```

diffusion_use_paquette ¶

if true, use atomic diffusion coefficients according to Paquette et al. (1986). if false, use Stanton & Murillo (2016) for diffusion coefficients. (Paquette coefficients still used for electron-ion because Stanton & Murillo did not do calculations for attractive potentials.)

```
diffusion_use_paquette = .false.
```

diffusion_use_iben_macdonald ¶

if true, use diffusion coefficients similar to Iben & MacDonald (1985). if false, use Stanton & Murillo (2016) for diffusion coefficients. this was previously called `diffusion_use_pure_coulomb`.

```
diffusion_use_iben_macdonald = .false.
```

diffusion_use_cgs_solver ¶

if false, solve the system of equations descibed by Thoul et al. (1994) if true, solve the unmodified Burgers equations in cgs units

```
diffusion_use_cgs_solver = .true.
```

cgs_thermal_diffusion_eta_full_on ¶

cgs_thermal_diffusion_eta_full_off ¶

When `diffusion_use_cgs_solver = .true.` for `eta < cgs_thermal_diffusion_eta_full_on`, includes the heat flow vector terms in the Burgers equations. Then smoothly turns off use of these terms so that they are not included for `eta > cgs_thermal_diffusion_eta_full_off`, since these terms are problematic when distribution function become non-Maxwellian.

```
cgs_thermal_diffusion_eta_full_on = 0d0
cgs_thermal_diffusion_eta_full_off = 2d0
```

do_Ne22_sedimentation_heating ¶

if true, include heating from sedimentation of Ne22 when element diffusion is on. Your net must include Ne22 for this to work. For best results, Ne22 should be treated as its own diffusion class. This will affect white dwarf cooling times. See also `eps_Ne22_sedimentation_factor`

```
do_Ne22_sedimentation_heating = .false.
```

diffusion_min_dq_at_surface ¶

treat at least this much at surface as a single cell for purposes of diffusion

```
diffusion_min_dq_at_surface = 1d-9
```

diffusion_min_T_at_surface ¶

treat cells cells at surface with T < this as a single cell for purposes of diffusion default should be large enough to ensure hydrogen ionization

```
diffusion_min_T_at_surface = 1d4
```

diffusion_min_dq_ratio_at_surface ¶

combine cells at surface until have total mass >= this factor times the next cell below them this helps with surface boundary condition for diffusion by putting large cell at surface

```
diffusion_min_dq_ratio_at_surface = 10
```

diffusion_dt_div_timescale ¶

dt is at most this fraction of timescale. Each stellar evolution step can be divided into many substeps for diffusion. The substep timescale is set by rates of flow in and out for each species in each cell. The substep size, dt, is initially set to `timescale*diffusion_dt_div_timescale`.

```
diffusion_dt_div_timescale = 1
```

diffusion_min_num_substeps ¶

Max substep dt is total time divided by this.

```
diffusion_min_num_substeps = 1
```

diffusion_max_iters_per_substep ¶

If the substep requires too many iterations, the substep time is decreased for a retry.

```
diffusion_max_iters_per_substep = 10
```

diffusion_max_retries_per_substep ¶

If the substep requires too many retries, diffusion fails and forces a retry for the star.

```
diffusion_max_retries_per_substep = 10
```

diffusion_tol_correction_max ¶

diffusion_tol_correction_norm ¶

Tolerances for newton iterations. Corrections smaller will be treated as converged. Corrections larger will cause another newton iteration.

```
diffusion_tol_correction_max = 1d-1
diffusion_tol_correction_norm = 1d-3
```

diffusion_min_X_hard_limit ¶

tolerance for negative mass fraction errors errors larger will cause retry; errors smaller will be corrected.

```
diffusion_min_X_hard_limit = -1d-3
```

diffusion_X_total_atol ¶

diffusion_X_total_rtol ¶

tolerances for errors in total species conservation errors larger will cause retry; errors smaller will be corrected.

```
diffusion_X_total_atol = 1d-9
diffusion_X_total_rtol = 1d-6
```

diffusion_upwind_abs_v_limit ¶

switch to upwind for i at face k if abs(v(i,k)) > this limit mainly for use with radiative levitation where get very much higher velocities

```
diffusion_upwind_abs_v_limit = 1d99
```

diffusion_v_max ¶

Max velocity (cm/sec). We can get extremely large velocities in the extreme outer envelope that cause problems numerically without really effecting the results, so we allow a max for the velocities that should help the numerics without changing the results. Note: change `diffusion_v_max` to at least 1d-2 when using radiative levitation.

```
diffusion_v_max = 1d-3
```

D_mix_ignore_diffusion ¶

Diffusion is turned off in core and surface convection zones, since it is overwhelmed by other mixing there. D_mix_ignore_diffusion roughly defines the mixing coefficient below which diffusion is included again. The code finds the location where D_mix falls to this value, backs up some, and turns on diffusion from there onward.

```
D_mix_ignore_diffusion = 1d5
```

diffusion_gamma_full_off ¶

diffusion_gamma_full_on ¶

`gamma_full_on <= gamma_full_off` Shut off diffusion for large gamma (i.e. for `gamma >= gamma_full_off`). Gradually decrease diffusion as gamma increases from `full_on` to `full_off`. Allow normal diffusion for `gamma <= gamma_full_on`. Default is diffusion off when get well into liquid regime.

```
diffusion_gamma_full_off = 175
diffusion_gamma_full_on = 150
```

diffusion_T_full_on ¶

diffusion_T_full_off ¶

`T_full_on >= T_full_off` Shut off diffusion for small T (i.e., for `T <= T_full_off`) Gradually decrease diffusion as T decreases from `T_full_on` to `T_full_off`. Allow normal diffusion for `T >= T_full_on`.

```
diffusion_T_full_on = 1d3
diffusion_T_full_off = 1d3
```

diffusion_calculates_ionization ¶

If `diffusion_calculates_ionization` is false, MESA uses typical charges for a set of representative species as defined in `diffusion_class_typical_charge` and `diffusion_class_representative` for all points rather than calculating the ionization from the local conditions.

```
diffusion_calculates_ionization = .true.
```

diffusion_nsmooth_typical_charge ¶

smoothing over charge

```
diffusion_nsmooth_typical_charge = 10
```

diffusion_SIG_factor ¶

diffusion_GT_factor ¶

factors for playing with SIG and GT terms for concentration diffusion and advection

```
diffusion_SIG_factor = 1d0
diffusion_GT_factor = 1d0
```

diffusion_AD_dm_full_on ¶

diffusion_AD_dm_full_off ¶

diffusion_AD_boost_factor ¶

artificial concentration diffusion near surface (mainly for radiative levitation) Msun units for `full_on` and `full_off` boost only used if > 0

```
diffusion_AD_dm_full_on = -1
diffusion_AD_dm_full_off = -1
diffusion_AD_boost_factor = 0
```

diffusion_Vlimit_dm_full_on ¶

diffusion_Vlimit_dm_full_off ¶

in Msun units artificial velocity limitation near surface (mainly for radiative levitation)

```
diffusion_Vlimit_dm_full_on = -1
diffusion_Vlimit_dm_full_off = -1
```

diffusion_Vlimit ¶

In units of local cell crossing velocity (only used if > 0). When full on, limit `abs(v) <= Vlimit*dr/dt`, cell size dr, substep time dt.

```
diffusion_Vlimit = 0
```

diffusion_min_T_for_radaccel ¶

diffusion_max_T_for_radaccel ¶

If T between these limits, then include radiative levitation at that location. Calculation of radiative levitation is costly, so only use it where necessary. Note: change `diffusion_v_max` to at least 1d-2 when using radiative levitation.

Note that radiative levitation requires OP calculations of g_rad for each class, and only 17 elements are supported (H, He, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Cr, Mn, Fe, Ni). If you want to include radiative levitation, your options are:

- Define diffusion classes such that all class representatives are among the 17 elements listed above.
- Use a net with only elements from the 17 above, and set diffusion_use_full_net = .true.

```
diffusion_min_T_for_radaccel = 0
diffusion_max_T_for_radaccel = 0
```

diffusion_min_Z_for_radaccel ¶

diffusion_max_Z_for_radaccel ¶

If Z between these limits, then include radiative levitation for that element. Calculation of radiative levitation is costly, so only use it where necessary. e.g., limit to Fe and Ni by `min_Z = 26` and `max_Z = 28`

```
diffusion_min_Z_for_radaccel = 0
diffusion_max_Z_for_radaccel = 1000
```

diffusion_screening_for_radaccel ¶

Include screening for radiative levitation.

```
diffusion_screening_for_radaccel = .true.
```

diffusion_use_full_net ¶

If true, don’t lump elements into classes for diffusion. Instead, each isotope in the network is treated as its own separate class. This can cause significant slowdowns for large nets, so it is off by default. This works for nets with up to 100 isotopes; larger nets require lumping into classes.

```
diffusion_use_full_net = .false.
```

diffusion_num_classes ¶

Number of representative classes of species for diffusion calculations. (maximum of 100)

```
diffusion_num_classes = 5
```

diffusion_class_representative(:) ¶

isotope names for diffusion representatives

```
diffusion_class_representative(1) = 'h1'
diffusion_class_representative(2) = 'he3'
diffusion_class_representative(3) = 'he4'
diffusion_class_representative(4) = 'o16'
diffusion_class_representative(5) = 'fe56'
```

diffusion_class_A_max(:) ¶

atomic number A. in ascending order. species goes into 1st class with `A_max` >= species A

```
diffusion_class_A_max(1) = 2
diffusion_class_A_max(2) = 3
diffusion_class_A_max(3) = 4
diffusion_class_A_max(4) = 16
diffusion_class_A_max(5) = 10000
```

diffusion_class_typical_charge(:) ¶

Typical charges for use if `diffusion_calculates_ionization` is false Use charge 21 for Fe in the sun, from Thoul, Bahcall, and Loeb (1994), ApJ, 421, 828.

```
diffusion_class_typical_charge(1) = 1
diffusion_class_typical_charge(2) = 2
diffusion_class_typical_charge(3) = 2
diffusion_class_typical_charge(4) = 8
diffusion_class_typical_charge(5) = 21
```

diffusion_class_factor(:) ¶

Arbitrarily enhance or inhibit diffusion effects by class.

```
diffusion_class_factor(:) = 1d0
```

parameters for ionization solver ¶

diffusion_use_isolve ¶

Activate iterative solver.

```
diffusion_use_isolve = .false.
```

diffusion_rtol_for_isolve ¶

diffusion_atol_for_isolve ¶

Relative and absolute error parameters for iterative solver.

```
diffusion_rtol_for_isolve = 1d-4
diffusion_atol_for_isolve = 1d-5
```

diffusion_maxsteps_for_isolve ¶

Maximum number of steps to take in iterative solver.

```
diffusion_maxsteps_for_isolve = 1000
```

diffusion_isolve_solver ¶

Which ode solver to use for iterative.

Options include:

- 'ros2_solver'
- 'rose2_solver'
- 'ros3p_solver'
- 'ros3pl_solver'
- 'rodas3_solver'
- 'rodas4_solver'
- 'rodasp_solver'

```
diffusion_isolve_solver = 'ros2_solver'
```

diffusion_dump_call_number ¶

debugging info of diffusion at call number

```
diffusion_dump_call_number = -1
```

eos controls ¶

more eos controls can be found in `star_job.defaults`

use_eosDT_ideal_gas ¶

if true, then eos is ideal gas eos as implemented by HELMEOS

```
use_eosDT_ideal_gas = .false.
```

use_eosDT_HELMEOS ¶

if true, then eos is as implemented by HELMEOS alone; no blending.

```
use_eosDT_HELMEOS = .false.
```

eosDT_HELMEOS_include_radiation ¶

if `use_eosDT_HELMEOS`, then this flag is passed as arg to control whether include radiation.

```
eosDT_HELMEOS_include_radiation = .true.
```

eosDT_HELMEOS_always_skip_elec_pos ¶

if `use_eosDT_HELMEOS`, then this flag is passed as arg to control whether skip electrons and positrons. if true, then always skip them else skip only for low T, low density situations

```
eosDT_HELMEOS_always_skip_elec_pos = .false.
```

eosDT_HELMEOS_always_include_elec_pos ¶

if `use_eosDT_HELMEOS`, then this flag is passed as arg to control whether skip electrons and positrons. if true, then always include them else include only for low T, low density situations

```
eosDT_HELMEOS_always_include_elec_pos = .false.
```

use_eosPTEH_for_low_density ¶

use_eosPTEH_for_high_Z ¶

Z_for_all_PTEH ¶

Z_for_any_PTEH ¶

overwrites the control of same name in the `Eos_General_Info` structure if use_eosPTEH_for_low_density is true, use PTEH for logRho < -10 or so. i.e., regions not covered by OPAL/SCVH. if use_eosPTEH_for_high_Z is true, use PTEH in place of HELM for Z above OPAL/SCVH range Z >= Z_for_all_PTEH, then use pure PTEH with no blend Z < Z_for_any_PTEH, then do not use PTEH blend with PTEH for intermediate Z. if use_eosDT2, blend with DT2. else blend with OPAL/SCVH data.

```
use_eosPTEH_for_low_density = .true.
use_eosPTEH_for_high_Z = .true.
Z_for_all_PTEH = 0.040d0
Z_for_any_PTEH = 0.039d0
```

use_eosDT2 ¶

overwrites the control of same name in the `Eos_General_Info` structure if use_eosPTEH_for_high_Z and use_eosDT2, use DT2 in place of OPAL/SCVH for cases with Z < Z_for_all_PTEH

```
use_eosDT2 = .false.
```

use_eosELM ¶

logT_max_for_ELM ¶

overwrites the control of same name in the `Eos_General_Info` structure if use_eosELM is true, use ELM in high Rho and T regions where otherwise use HELM. limit use of ELM to logT <= logT_max_for_ELM

```
use_eosELM = .false.
logT_max_for_ELM = 8.8d0
```

use_fixed_XZ_for_eos ¶

for debugging

```
use_fixed_XZ_for_eos = .false.
```

fixed_X_for_eos ¶

if `use_fixed_XZ_for_eos`, then pass this value to eos instead of actual X.

```
fixed_X_for_eos = -1
```

fixed_Z_for_eos ¶

if `use_fixed_XZ_for_eos`, then pass this value to eos instead of actual Z.

```
fixed_Z_for_eos = -1
```

opacity controls ¶

more opacity controls can be found in `star_job.defaults`

cubic_interpolation_in_X ¶

type of interpolation in X

```
cubic_interpolation_in_X = .false.
```

cubic_interpolation_in_Z ¶

type of interpolation in Z

```
cubic_interpolation_in_Z = .false.
```

include_electron_conduction ¶

add conduction opacities to radiative opacities

```
include_electron_conduction = .true.
```

use_simple_es_for_kap ¶

for experiments with simple electron scattering if true, `opacity = 0.2*(1 + x)`

```
use_simple_es_for_kap = .false.
```

use_Type2_opacities ¶

Type2 opacities for extra C/O during and after He burning. To use Type2 opacities one needs to specify a base metallicity, Zbase, which gives the metal abundances previous to any CO enhancement. In regions where central hydrogen is above a given threshold, or the metallicity is not significantly higher than Zbase, Type1 tables are used instead, with blending regions to smoothly transition from one to the other. Why not just use Type2 all the time? Is it a performance reason to still support Type1s?

No – the Type1 tables cover a wider range of X and have a higher resolution in Z for each X.

The Type1 tables are for (X,Z) pairs from the following sets:

The 10 Type1 X's are 0.0, 0.1, 0.2, 0.35, 0.5, 0.7, 0.8, 0.9, 0.95, 1-Z The 13 Type1 Z's are 0.0, 1e-4, 3e-4, 1e-3, 2e-3, 4e-3, 1e-2, 2e-2, 3e-2, 4e-2, 6e-2, 8e-1, 1e-1

The Type2 tables are for (X,Z) pairs from the following more limits sets: The 5 Type2 X's are 0.0, 0.03, 0.10, 0.35, 0.70 The 8 Type2 Z's are 0.00, 0.001, 0.004, 0.01, 0.02, 0.03, 0.05, 0.1

There are 130 (X,Z) combinations for Type1 and only 40 for Type2. So Type2 gives you C/O enhancement at a cost of lower resolution in (X,Z). The Type1 tables also cover the full possible range of X from 0.0 to 1-Z, whereas the Type2 tables stop at a max X of 0.70. Extrapolating the Type2 tables to higher X is not reliable, so we switch over to using Type1 data instead. In this case, Type1 opacities are computed using Zbase instead of the actual metallicity.

```
use_Type2_opacities = .false.
```

Zbase ¶

the base metallicity for the Type2 kap evaluations.

```
Zbase = -1
```

use_Zbase_for_Type1_blend ¶

If true, then if `use_Type2_opacities = .true.` Type1 opacities will be computed using Zbase instead of Z. Ignored if `use_Type2_opacities = .false.`

```
use_Zbase_for_Type1_blend = .true.
```

kap_Type2_full_off_X ¶

kap_Type2_full_on_X ¶

switch to Type1 if X too large Type2 is full off for `X >= kap_Type2_full_off_X` Type2 can be full on for `X <= kap_Type2_full_on_X`

```
kap_Type2_full_off_X = 0.71d0
kap_Type2_full_on_X = 0.70d0
```

kap_Type2_full_off_dZ ¶

kap_Type2_full_on_dZ ¶

switch to Type1 if dZ too small `(dZ = Z - Zbase)` Type2 is full off for `dZ <= kap_Type2_full_off_dZ` Type2 can be full on for `dZ >= kap_Type2_full_on_dZ`.

```
kap_Type2_full_off_dZ = 0.001d0
kap_Type2_full_on_dZ = 0.01d0
```

X and dZ terms are multiplied to get actual fraction of Type2. The fraction of Type2 is calculated for each cell depending on the X and dZ for that cell. So you can be using Type1 in cells where X is large or dZ is small, while at the same time you can be using Type2 where X is small and dZ is large. When `frac_Type2` is > 0 and < 1, then both Type1 and Type2 are evaluated and combined linearly as `(1-frac_Type2)*kap_type1 + frac_Type2*kap_type2`. Add `kap_frac_Type2` to your profile columns list to see `frac_Type2` for each cell.

opacity_max ¶

limit opacities to this value (ignore this is value is < 0)

```
opacity_max = -1
```

opacity_factor ¶

opacities are multiplied by this value

```
opacity_factor = 1
```

min_logT_for_opacity_factor_off ¶

min_logT_for_opacity_factor_on and ¶

max_logT_for_opacity_factor_on ¶

max_logT_for_opacity_factor_off ¶

temperature controls for where the `opacity_factor` is applied if, for example, you only want the opacity factor to apply in the iron bump region you can give a logT range such as

```
min_logT_for_opacity_factor_off = 5.2
min_logT_for_opacity_factor_on = 5.3
max_logT_for_opacity_factor_on = 5.7
max_logT_for_opacity_factor_off = 5.8
```

ignore these if < 0.

```
min_logT_for_opacity_factor_off = -1
min_logT_for_opacity_factor_on = -1
max_logT_for_opacity_factor_on = -1
max_logT_for_opacity_factor_off = -1
```

if you need cell-by-cell control of opacity factor, set the vector "`extra_opacity_factor`" using the routine "`other_opacity_factor`"

OP mono opacities ¶

The `OP_mono` opacities use data and code from the OP website as modified by Haili Hu. Since the tar.gz file is large (656 MB), it is not included in the standard mesa download.

You can get `OP4STARS_1.3.tar.gz` here

Put it any place you want on your disk.

```
gunzip OP4STARS_1.3.tar.gz
tar -xvf OP4STARS_1.3.tar
```

Set the inlist controls for the "mono" directory with the data files. For example, in my case it looks like the following, but you can put the directory anywhere you like – it doesn't need to be in the mesa/data directory. And the cache file doesn't need to be in the mono directory.

```
op_mono_data_path = '/Users/bpaxton/OP4STARS_1.3/mono'
op_mono_data_cache_filename = '/Users/bpaxton/OP4STARS_1.3/mono/op_mon
```

op_mono_data_path ¶

if this path is set to the empty string, '', then it defaults to the environment variable `$(MESA_OP_MONO_DATA_PATH)`

```
op_mono_data_path = ''
```

op_mono_data_cache_filename ¶

if this is set to the empty string, ‘’, then it defaults to the environment variable $(MESA_OP_MONO_DATA_CACHE_FILENAME)

```
op_mono_data_cache_filename = ''
```

high_logT_op_mono_full_off ¶

high_logT_op_mono_full_on ¶

low_logT_op_mono_full_off ¶

low_logT_op_mono_full_on ¶

you can select a range of log10T for using op_mono opacities outside that range, the code will use standard opacity tables. for example, you might only use high T limits so that op_mono is only used in the envelope, or you might set both low and high T limits so that op_mono is used around the Fe peak logT but not for other locations in the star.

```
high_logT_op_mono_full_off >= high_logT_op_mono_full_on
high_logT_op_mono_full_on >= low_logT_op_mono_full_on
low_logT_op_mono_full_on >= low_logT_op_mono_full_off

op_mono opacities full on if
log10T <= high_logT_op_mono_full_on
and
log10T >= low_logT_op_mono_full_on

op_mono opacities full off if
log10T >= high_logT_op_mono_full_off
or
log10T <= low_logT_op_mono_full_off
```

partially on for other cases

```
high_logT_op_mono_full_off = -1d99
high_logT_op_mono_full_on = -1d99
```

```
low_logT_op_mono_full_off = -1d99
low_logT_op_mono_full_on = -1d99
```

op_mono_min_X_to_include ¶

skip iso if mass fraction < this

```
op_mono_min_X_to_include = 1d-20
```

use_op_mono_alt_get_kap ¶

if true, call the `op_mono_alt_get_kap` routine instead of `op_mono_get_kap`. see `mesa/kap/public/kap_lib.f` for details about these routines.

```
use_op_mono_alt_get_kap = .false.
```

kap_phot_factor_for_kap_floor ¶

kap_phot_step_factor ¶

min_kap_floor_step_factor ¶

min_for_kap_floor ¶

tau_use_kap_floor ¶

```
kap_phot_factor_for_kap_floor = 1d0
kap_phot_step_factor = 0.05d0
min_kap_floor_step_factor = 0.333d0
min_for_kap_floor = 1d-4
tau_use_kap_floor = .false.
```

kap_min_Z_0pt02 ¶

kap_min_Z_1pt0 ¶

```
kap_min_Z_0pt02 = 1d-20
kap_min_Z_1pt0 = 1d-20
```

kap_fac_X_lo ¶

kap_fac_X_lo_fac ¶

kap_fac_X_hi ¶

kap_fac_X_hi_fac ¶

```
kap_fac_X_lo = -1
kap_fac_X_lo_fac = -1
kap_fac_X_hi = -1
kap_fac_X_hi_fac = -1
```

asteroseismology controls ¶

get_delta_nu_from_scaled_solar ¶

use scaled solar values

```
get_delta_nu_from_scaled_solar = .false.
```

nu_max_sun ¶

solar value of `nu_max`

```
nu_max_sun = 3100d0
```

delta_nu_sun ¶

solar value of `delta_nu`

```
delta_nu_sun = 135d0
```

Teff_sun ¶

solar value of Teff

```
Teff_sun = 5777d0
```

delta_Pg_mode_freq ¶

uHz. if <=0, use nu_max from scaled solar value

```
delta_Pg_mode_freq = 0d0
```

Brunt controls ¶

calculate_Brunt_N2 ¶

Only calculate `Brunt_N2` if this is true.

```
calculate_Brunt_N2 = .true.
```

brunt_N2_coefficient ¶

Standard N2 is multiplied by this value.

```
brunt_N2_coefficient = 1
```

num_cells_for_smooth_brunt_B ¶

Number of cells on either side to use in weighted smoothing of `brunt_B`.

```
num_cells_for_smooth_brunt_B = 2
```

threshold_for_smooth_brunt_B ¶

Threshold for weighted smoothing of `brunt_B`. Only apply smoothing (controlled by `num_cells_for_smooth_bruntB`) for contiguous regions where |bruntB| exceeds this threshold. Might be useful for preventing narrow peaks from being excessively broadened by smoothing

```
threshold_for_smooth_brunt_B = 0d0
```

use_brunt_gradmuX_form ¶

For comparison to older codes. Assumes ideal gas plus radiation for `brunt_B`. Uses hydrogren mass fraction to estimate dlnmu = dX/(X + 0.6).

```
use_brunt_gradmuX_form = .false.
```

interpolate_rho_for_pulsation_info ¶

If true, then get `rho_face` by interpolating rho at cell center. If false, then calculate `rho_face` by `dm/(4*pi*r^2*dr)`.

```
interpolate_rho_for_pulsation_info = .true.
```

min_magnitude_brunt_B ¶

If set `brunt_B` to 0 if absolute value is < this.

```
min_magnitude_brunt_B = -1d99
```

structure equations ¶

velocity_q_upper_bound ¶

Local override for global `v_flag`. If local q > this bound, local `v_flag` is set false, else local `v_flag` is set to global `v_flag`. this lets you force v = 0 in outer envelope.

```
velocity_q_upper_bound = 1d99
```

velocity_logT_lower_bound ¶

Local override for global `v_flag`. If local logT < this bound, local `v_flag` is set false, else local `v_flag` is set to global `v_flag`. this lets you force v = 0 in outer envelope.

```
velocity_logT_lower_bound = -1d99
```

sponge_max_q_full_on ¶

sponge_min_q_full_off ¶

"sponge" soaks up velocities; full on mean velocities forced to zero. sponge full on for `q >= sponge_max_q_full_on` (== velocities full off) sponge full off for `q <= sponge_min_q_full_off` `sponge_min_q_full_off < sponge_max_q_full_on`

```
sponge_max_q_full_on = -1d99
sponge_min_q_full_off = -1d99
```

max_dt_yrs_for_velocity_logT_lower_bound ¶

Only apply `velocity_logT_lower_bound` when timestep < this limit.

```
max_dt_yrs_for_velocity_logT_lower_bound = 1d99
```

use_dP_dm_rotation_correction ¶

With rotation, multiply dP/dm by `fp_rot` if this flag is true.

```
use_dP_dm_rotation_correction = .true.
```

use_mass_corrections ¶

Gravitational vs baryonic mass corrections. If false, then no distinction between gravitational and baryonic mass. If true, then gravitational mass is calculated using mass corrections. Note: may need to wait for pre-ms model to converged before turning this on.

```
use_mass_corrections = .false.
```

use_sr_sound_speed ¶

SR correction for sound speed.

```
use_sr_sound_speed = .false.
```

use_gr_factors ¶

GR corrections. Currently just for pressure equation.

```
use_gr_factors = .false.
```

use_ODE_var_eqn_pairing ¶

changes the pairing of equations and variables helps with numerical issues in hydro matrix solves

```
use_ODE_var_eqn_pairing = .false.
```

use_dvdt_form_of_momentum_eqn ¶

if true, use dv/dt = … form of momentum equation. this replaces the default pressure gradient form. only when v_flag is true.

```
use_dvdt_form_of_momentum_eqn = .false.
```

use_Paczynski_term_in_dvdt_eqn ¶

if true, then the dv/dt = … form of momentum equation. includes Paczynski correction term in optically thin regions. B. Paczynski, 1969, Acta Astr., vol. 19

```
use_Paczynski_term_in_dvdt_eqn = .false.
```

use_dedt_form_of_energy_eqn ¶

if true, use de/dt = … form of energy equation. this replaces the default dL/dm and eps_grav form.

```
use_dedt_form_of_energy_eqn = .false.
```

use_ODE_form_of_density_eqn ¶

if true, use ODE dlnd/dt = … if false, use algebraic relation `rho = cell_mass/cell_vol`

```
use_ODE_form_of_density_eqn = .false.
```

non_nuc_neu_factor ¶

Multiplies power from non-nuclear reaction neutrinos. i.e., thermal neutrinos such as computed by mesa/neu.

```
non_nuc_neu_factor = 1
```

eps_nuc_factor ¶

Multiplies eps_nuc without changing rates or dxdt_nuc. Thus controls energy production without modifying the amount of change in abundances.

```
eps_nuc_factor = 1
```

eps_Ne22_sedimentation_factor ¶

This controls energy production from element diffusion sedimentation of Ne22.

```
eps_Ne22_sedimentation_factor = 1
```

max_abs_eps_nuc ¶

Limit magnitude of eps_nuc to this.

```
max_abs_eps_nuc = 1d99
```

fe56ec_fake_factor ¶

min_T_for_fe56ec_fake_factor ¶

Multiplier on ni56 electron capture rate to take isotopes in hardwired networks to more neutron rich isotopes.

```
fe56ec_fake_factor = 1d-7
min_T_for_fe56ec_fake_factor = 3d9
```

eps_grav ¶

In mesa, "eps_grav" means -T*dS/dt which is equivalent to -(dE/dt + P*dV/dt) where S is specific entropy, E is specific internal energy, and V = 1/rho. There are several options for how eps_grav is calculated. These alternatives are equivalently from the "ideal" physics viewpoint, but they can be very different numerically depending on the situation.

The standard default forms are used if you don't set any of the following flags. The default when using lnd instead of lnPgas as primary variable is

```
eps_grav = -T*cp*((1-grada)*chiT*dlnT_dt - grada*chiRho*dlnd_dt)
```

When using lnPgas instead of lnd, the default is

```
eps_grav = -T*cp((1-grada*4*Prad/P)*dlnT_dt - grada*Pgas/P*dlnPgas_dt)
```

use_dEdRho_form_for_eps_grav ¶

If true, use `eps_grav = -(cv*T*dlnT_dt + (rho*dE_dRho - P/rho)*dlnd_dt)`

```
use_dEdRho_form_for_eps_grav = .false.
```

use_dlnd_dt_form_for_eps_grav ¶

If true, use `eps_grav = -dedt + P/rho*dlnd_dt`.

```
use_dlnd_dt_form_for_eps_grav = .false.
```

use_PdVdt_form_for_eps_grav ¶

If true, use `eps_grav = -(dedt + P*d(1/rho)/dt). [1/rho = V,specific vol.]` With time centering for 1/rho, `P*d(1/rho)/dt becomes P*(1/rho - 1/rho_start)/dt`

```
use_PdVdt_form_for_eps_grav = .false.
```

use_lnS_for_eps_grav ¶

If true, use `eps_grav = -T*DS/Dt`. Note: while this seems like the obvious way to go, it has problems numerically. lnS is not a basic variable in the way that lnT and lnd (or lnPgas) are. I.e., we get lnT and lnd from the newton solver directly, but we get lnS by calling the eos using lnT and lnd as args. Also, in many cases, the cell mass coordinates don't change for the step, making it possible to use the solver value for the increment in the lnT or lnd directly in estimating the Lagrangian time derivative (e.g., DlnT/dt = /dt instead of = (new lnT - old lnT)/dt). By avoiding the roundoff error in this way, we also get a boost numerically. With lnS we cannot do that -- we're stuck with DlnS/Dt = (new lnS - old lnS)/dt. In a perfect world with infinite precision, none of this would matter. But in practice, it turns out to make a significant difference, so unless there are strong reasons otherwise, you should use one of the other schemes for getting `eps_grav`. One situation where you need to use the lnS version is deep in a white dwarf where you can have a phase transition that the lnS form will take care of but the others won't. That particular case is handled using `Gamma_lnS_eps_grav_full_off`/on.

```
use_lnS_for_eps_grav = .false.
```

include_dmu_dt_in_eps_grav ¶

The above do not include the contribution from composition changes. In most cases, that is okay (at least it is a common practice!), but for high T, high density situations, you may want to full the full form. to do that, set `include_dmu_dt_in_eps_grav` to true.

This only is relevant when you are not using the lnS form of `eps_grav`. Since when using `eps_grav = -T*dS/dt`, the composition effects are already included. otherwise, we calculate the composition term in `eps_grav` as `-dE_dmu*dmu_dt` with mu approximated by `abar/(1 + zbar)` corresponding to complete ionization and `dE_dmu` approximated by `-3/2*cgas*T/mu^2` (cgas = ideal gas constant; erg/K/mole) where `dmu_dt` is 1st order approximation Lagrangian time derivative, `(mu - prev_mu)/dt`, `prev_mu` interpolated at same mass coordinate in start-of-step model.

```
include_dmu_dt_in_eps_grav = .false.
```

Gamma_lnS_eps_grav_full_off ¶

Gamma_lnS_eps_grav_full_on ¶

Automatic switch to lnS form for regions with high Gamma (plasma interaction parameter). Set `use_lnS_for_eps_grav` false to use these controls. These are ignored when `use_lnS_for_eps_grav` is true.

```
Gamma_lnS_eps_grav_full_on = 150d0
Gamma_lnS_eps_grav_full_off = 120d0
```

eps_grav_factor ¶

multiply eps_grav by this factor

```
eps_grav_factor = 1
```

eps_grav_dt_use_start_values ¶

set true if must use values for lnT, lnd, or lnP from start of step in d_dt. e.g., if have made significant changes in abundance profiles in diffusion.

```
eps_grav_dt_use_start_values = .false.
```

eps_grav_time_deriv_separation ¶

Separation (in grid cells) over which eps_grav can be time-differenced when Mstar changes The mesh has two major regions - an interior region where the cells are Lagrangian and an outer region where they are homologous (constant dq =dm/M). There is also a small transition region between these two. In the Lagrangian region Ds/dt is evaluated with a Lagrangian finite difference in time, while in the homologous region Ds/dt is evaluated with a finite difference in time at constant q plus an advection-like term accounting for the movement of the q boundaries in mass. In the transition region, these two derivatives are combined. This means that at the edges of the transition region, finite differences may cross cell boundaries. This control determines how the mesh for the end of the current timestep is placed to ensure that finite differences cross no more than this many cell boundaries.

```
eps_grav_time_deriv_separation = 1.5d0
```

zero_eps_grav_in_just_added_material ¶

If true, set `eps_grav(k) = 0` for `k < k_below_just_added`. NOTE: this does not mean that Ds/Dt is forced to be 0 in cell k. Instead it simply means we ignore Ds/Dt in the cell's energy calculation.

```
zero_eps_grav_in_just_added_material = .false.
```

min_dxm_Eulerian_div_dxm_removed ¶

Controls for Eulerian or Lagrangian forms of `eps_grav`. Only for mass loss. Specifies a minimum value for the ratio of the mass layer at the surface using Eulerian `eps_grav` (`dxm_Eulerian`) divided by the mass removed in the current step.

```
min_dxm_Eulerian_div_dxm_removed = 2
```

min_dxm_Eulerian_div_dxm_added ¶

Controls for Eulerian or Lagrangian forms of `eps_grav`. Only for mass gain. Specifies a minimum value for the ratio of the mass layer at the surface using Eulerian `eps_grav` (`dxm_Eulerian`) divided by the mass added in the current step.

```
min_dxm_Eulerian_div_dxm_added = 5
```

min_cells_for_Eulerian_to_Lagrangian_transition ¶

Width of eulerian to lagrangian transition region.

```
min_cells_for_Eulerian_to_Lagrangian_transition = 10
```

fix_eps_grav_transition_to_grid ¶

If true, fix the transition region for the computation of `eps_grav` to the transition from Lagrangian to constant in q of the grid.

```
fix_eps_grav_transition_to_grid = .false.
```

min_del_T_div_dt ¶

Controls for Lagrangian time derivatives in newly added material only applies to cells with `k < k_below_just_added`. If `del_t_for_just_added(k)/dt < this limit`, then `set del_t_for_just_added(k) = dt*this limit`.

```
min_del_T_div_dt = 1d-10
```

max_num_surf_revisions ¶

Max number of forced reconverges for changes in `surf_lnS`.

```
max_num_surf_revisions = 1
```

max_abs_rel_change_surf_lnS ¶

Force newton reconverge if surf_lnS changed more than this.

```
max_abs_rel_change_surf_lnS = 5d-4
```

trace_force_another_iteration ¶

If true, report when force another iter.

```
trace_force_another_iteration = .false.
```

accel_factor ¶

coefficient for acceleration term in the momentum equation

```
accel_factor = 1
```

extra_power_source ¶

erg/g/sec applied uniformly throughout the model This can be used to push a pre-ms model up the track to lower center temperatures. Can be used simultaneously with `inject_extra_ergs_sec` and `inject_uniform_extra_heat`

```
extra_power_source = 0
```

inject_uniform_extra_heat ¶

extra heat in erg g^-1 s^-1 Added to cells in range `min_q_for_uniform_extra_heat` to max. Can be used simultaneously with `inject_extra_ergs_sec` and `extra_power_source`.

```
inject_uniform_extra_heat = 0
```

min_q_for_uniform_extra_heat ¶

sets bottom of region for `inject_uniform_extra_heat`

```
min_q_for_uniform_extra_heat = 0
```

max_q_for_uniform_extra_heat ¶

sets top of region for `inject_uniform_extra_heat`

```
max_q_for_uniform_extra_heat = 1
```

inject_extra_ergs_sec ¶

added to mass equal to `grams_for_inject_extra_core_ergs_sec` can be used simultaneously with `extra_power_source` and `inject_uniform_extra_heat`

```
inject_extra_ergs_sec = 0
```

base_of_inject_extra_ergs_sec ¶

(units: Msun) sets bottom of region for `inject_extra_ergs_sec` note: actual base is at max of this and the center of the model

```
base_of_inject_extra_ergs_sec = 0
```

total_mass_for_inject_extra_ergs_sec ¶

(units: Msun) sets size of region for `inject_extra_ergs_sec`

```
total_mass_for_inject_extra_ergs_sec = 0
```

start_time_for_inject_extra_ergs_sec ¶

(units: sec) start time for injecting extra ergs/s

```
start_time_for_inject_extra_ergs_sec = -1d99
```

duration_for_inject_extra_ergs_sec ¶

(units: sec) length of time for injecting extra ergs/s set to negative value to keep injecting indefinitely or until reach target

```
duration_for_inject_extra_ergs_sec = -1
```

inject_until_reach_model_with_total_energy ¶

(units: ergs) target for model total energy usually want to set `duration_for_inject_extra_ergs_sec = -1` for this option. see also: `inject_until_reach_delta_total_energy` continue injecting until total energy of model reaches min of `inject_until_reach_model_with_total_energy`, and `inject_until_reach_delta_total_energy + initial total energy`

```
inject_until_reach_model_with_total_energy = 1d99
```

inject_until_reach_delta_total_energy ¶

(units: ergs) target for change in total energy stop injecting when `total_energy - total_energy_initial > this`. usually want to set `duration_for_inject_extra_ergs_sec = -1` for this option. see also: `inject_until_reach_model_with_total_energy` continue injecting until total energy of model reaches min of `inject_until_reach_model_with_total_energy`, and `inject_until_reach_delta_total_energy + initial total energy`

```
inject_until_reach_delta_total_energy = 1d99
```

max_inject_velocity_km_per_sec ¶

```
max_inject_velocity_km_per_sec = 0
```

theta_P ¶

for time weighting P in energy and momentum equations

```
<P> = theta_P*P + (1 - theta_P)*P_start
```

```
theta_P = 1d0
```

use_porosity_with_dPrad_dm_form ¶

```
use_porosity_with_dPrad_dm_form = .false.
```

qmax_zero_non_radiative_luminosity ¶

qmin_freeze_non_radiative_luminosity ¶

use_dPrad_dm_form_of_T_gradient_eqn ¶

use_flux_limiting_with_dPrad_dm_form ¶

These are for alternatives ways to determine the T gradient. The standard form of the equation is

```
dT/dm = dP/dm * T/P * grad_T, grad_T = dlnT/dlnP from MLT.
```

use hydrostatic value for dP/dm in this. this is because of limitations of MLT for calculating grad_T. (MLT assumes hydrostatic equilibrium) see comment in K&W chpt 9.1.

The alternatives forms are for dynamic situations where the use of hydrostatic dP/dm is inappropriate. In order of priority,

```
if q(k) > qmin_freeze_non_radiative_luminosity then
    use L_conv from start of step to get L_rad = L - L_conv_start
else if q(k) <= qmax_zero_non_radiative_luminosity then
    simply use L_rad = L
else if (use_dPrad_dm_form_of_T_gradient_eqn)
    if (gradT < gradr) then
       use L_rad = L*gradT/gradr (see, e.g., Cox&Giuli 14.109)
    else
       use L_rad = L
```

With the resulting `L_rad`, determine the expected dT/dm by

```
d_Prad/dm = -kap*L_rad/(clight*area^2) -- see, e.g., K&W (5.12)
```

```
qmax_zero_non_radiative_luminosity = 0d0
qmin_freeze_non_radiative_luminosity = 1d0
use_dPrad_dm_form_of_T_gradient_eqn = .false.
use_flux_limiting_with_dPrad_dm_form = .false.
```

center_energy_pulses ¶

pulse_step_inteval ¶

pulse_max_step ¶

pulse_ergs_even_steps ¶

pulse_ergs_odd_steps ¶

```
center_energy_pulses = .false.
pulse_step_inteval = 11
pulse_max_step = 1000
pulse_ergs_even_steps = 1d20
pulse_ergs_odd_steps = -1d20
```

for hydro comparison tests (e.g., Sedov)

gamma_law_hydro ¶

off as long as value is <= 0

```
gamma_law_hydro = 0d0
```

zero_gravity ¶

if true, then set G to zero

```
zero_gravity = .false.
```

disable_riemann_reconstruction ¶

if true, then set just use bounding cell average P and u at face

```
disable_riemann_reconstruction = .false.
```

constant_L ¶

if true, then L(k) = L_center for all k. disable dlnTdm equation.

```
constant_L = .false.
```

drag_per_step ¶

drag_per_second ¶

```
drag_per_step = -1d0
drag_per_second = -1d0
```

Rayleigh-Taylor Instability ¶

RTI_A ¶

RTI_B ¶

RTI_C ¶

RTI_D ¶

RTI_C_X_factor ¶

RTI_C_X0 ¶

RTI_max_alpha ¶

RTI_min_dm_behind_shock_for_full_on ¶

RTI_dm_for_center_alpha_nondecreasing ¶

RTI_energy_floor ¶

RTI_D_mix_floor ¶

RTI_min_m_for_D_mix_floor ¶

RTI_log_max_boost ¶

RTI_m_full_boost ¶

RTI_m_no_boost ¶

Note that these parameters are not exactly the same ¶

as used by Paul Duffell. ¶

His calibrated D is 2, where mesa uses D = 3 (see mesaIV paper). ¶

Users should try both values since the choice is not clear cut. ¶

```
RTI_A = 1d-3
RTI_B = 2.5d0
RTI_C = 0.2d0
RTI_D = 3d0
```

```
RTI_C_X0_frac = 0.9d0
RTI_C_X_factor = 0d0
```

```
RTI_max_alpha = 0.5d0
RTI_min_dm_behind_shock_for_full_on = 0d0
RTI_dm_for_center_eta_nondecreasing = 0.02d0
RTI_energy_floor = 0d0
RTI_D_mix_floor = 0d0
RTI_min_m_for_D_mix_floor = 0d0
```

```
RTI_log_max_boost = 3d0
RTI_m_full_boost = 4d0
RTI_m_no_boost = 5d0
```

solver controls ¶

the following is from a response on mesa-users to a question about controls for solver tolerances:

The “residual” is the left over difference between the left and right hand sides of the equation we are trying to solve. We do iterations to reduce that, but we are limited by the non-linearity of the problem and the quality of the estimates for the derivatives.

The “correction” is the change in the primary variable that is calculated using good-old Newton’s rule in multiple dimensions — so Jacobian and residuals give a correction that would make the next residual vanish

if the problem were linear and the Jacobian was exact, neither of which are true. So the best we can hope for is that the corrections will get smaller next time.

The “norm” is the average; the “max” is the max. Sometimes you mainly care about the norm and will accept a few outliers. But sometimes you don’t want any really bad outliers, so you want to set a low limit for the max residual or correction as well as the norm.

You might want to try for several iterations with strict tolerances, and then relax them if things are still not converged. For example, you might be willing to live with the larger tolerances, but you’d like to give it a good try at the smaller ones before switching. Also, you might be willing to settle for any-old residual if the corrections have become small enough. You can do that too by relaxing the residual tolerances after a few iterations.

Hope that at least helps with the nomenclature.

I agree with Frank that you should consider the effects of smaller timesteps and more grid points as your main technique — tightening up the tolerances for the solver won’t help if you are taking timesteps that are too large or if you have inadequate grid resolution.

tol_correction_norm ¶

tol_max_correction ¶

“Correction” for variable x(i,k) is scaled change, dx(i,k)/xscale(i,k). these tolerances are for the magnitude of the scaled corrections.

```
tol_correction_norm = 3d-5
tol_max_correction = 3d-3
```

tol_correction_high_T_limit ¶

For very late stages of massive star evolution, need to relax tolerances. If max T >= this limit, switch scaling factors.

```
tol_correction_high_T_limit = 1d9
```

tol_correction_norm_high_T ¶

tol_max_correction_high_T ¶

Above `tol_correction_high_T_limit` use these scaling factors.

```
tol_correction_norm_high_T = 3d-3
tol_max_correction_high_T = 3d-1
```

tol_correction_extreme_T_limit ¶

For very late stages of massive star evolution, need to relax tolerances. If center T >= this limit, switch scaling factors.

```
tol_correction_extreme_T_limit = 6d9
```

tol_correction_norm_extreme_T ¶

tol_max_correction_extreme_T ¶

For very late stages of massive star evolution, need to relax tolerances. If center T >= this limit, switch scaling factors.

```
tol_correction_norm_extreme_T = 8d-3
tol_max_correction_extreme_T = 8d-1
```

tol_bad_max_correction ¶

if `max_correction > tol_max_correction` and no more iterations allowed, then still accept the solution if `max_correction <= tol_bad_max_correction`. but if `max_correction > tol_bad_max_correction`, then reject the solution.

```
tol_bad_max_correction = 0d0
```

bad_max_correction_series_limit ¶

If have this many steps in a row with `max_correction > tol_max_correction`, then do a retry with a smaller timestep.

```
bad_max_correction_series_limit = 2
```

relax_use_gold_tolerances ¶

```
relax_use_gold_tolerances = .false.
```

relax_newton_iterations_limit ¶

relax_newton_iterations_hard_limit ¶

relax_tol_correction_norm ¶

relax_tol_max_correction ¶

relax_tol_residual_norm1 ¶

relax_tol_max_residual1 ¶

relax_iter_for_resid_tol2 ¶

relax_tol_residual_norm2 ¶

relax_tol_max_residual2 ¶

relax_iter_for_resid_tol3 ¶

relax_tol_residual_norm3 ¶

relax_tol_max_residual3 ¶

relax_maxT_for_gold_tolerances ¶

relax_max_eosPC_frac_for_gold_tolerances ¶

For use during relax operations. Only used if /= 0.

```
relax_newton_iterations_limit = 0
relax_newton_iterations_hard_limit = 0
```

```
relax_tol_correction_norm = 0d0
relax_tol_max_correction = 0d0
```

```
relax_tol_residual_norm1 = 0d0
relax_tol_max_residual1 = 0d0
relax_iter_for_resid_tol2 = 3
```

```
relax_tol_residual_norm2 = 0d0
relax_tol_max_residual2 = 0d0
relax_iter_for_resid_tol3 = 0
```

```
relax_tol_residual_norm3 = 0d0
relax_tol_max_residual3 = 0d0
relax_maxT_for_gold_tolerances = -1d0
relax_max_eosPC_frac_for_gold_tolerances = 0d0
```

include_L_in_error_est ¶

include_v_in_error_est ¶

include_u_in_error_est ¶

Some variables can be excluded from calculation of correction norm and max.

```
include_L_in_error_est = .false.
include_v_in_error_est = .false.
include_u_in_error_est = .false.
```

tol_correction_norm_alt ¶

tol_max_correction_alt ¶

If you have several backups in a row, your run is having a near death experience. So as a last hope, try relaxing the correction tolerances. It might help. The code will use these tolerances after 3 or more backups in a row. Once there is a step without a backup, it goes back to the normal tolerances.

```
tol_correction_norm_alt = 1d-3
tol_max_correction_alt = 1d-2
```

correction_xa_limit ¶

Ignore correction to abundance when calculating correction norm and max if current mass fraction is less than this limit.

```
correction_xa_limit = 5d-3
```

xa_scale ¶

Scaling for abundance variables is `max(xa_scale, current mass fraction)`.

```
xa_scale = 1d-5
```

tol_residual_norm1 ¶

tol_max_residual1 ¶

iter_for_resid_tol2 ¶

“residual” for equation is the difference between left and right sides use `tol_residual_norm1` & `tol_max_residual1` at iteration number `iter_for_resid_tol2`, switch to next tolerances.

```
tol_residual_norm1 = 1d-10
tol_max_residual1 = 1d-9
iter_for_resid_tol2 = 6
```

tol_residual_norm2 ¶

tol_max_residual2 ¶

iter_for_resid_tol3 ¶

Use `tol_residual_norm2` & `tol_max_residual2` these apply starting at iteration number `iter_for_resid_tol2`. at iteration number `iter_for_resid_tol3`, switch to next tolerances.

```
tol_residual_norm2 = 1d99
tol_max_residual2 = 1d99
iter_for_resid_tol3 = 15
```

tol_residual_norm3 ¶

tol_max_residual3 ¶

Use `tol_residual_norm3` & `tol_max_residual3` these apply starting at iteration number `iter_for_resid_tol3`.

```
tol_residual_norm3 = 1d99
tol_max_residual3 = 1d99
```

If things get worse from one iteration to next, give up. The following are the limits that define “getting worse enough to stop”.

corr_norm_jump_limit ¶

If correction norm increases by this factor or more, quit.

```
corr_norm_jump_limit = 1d99
```

max_corr_jump_limit ¶

If correction max increases by this factor or more, quit.

```
max_corr_jump_limit = 1d6
```

resid_norm_jump_limit ¶

If residual norm increases by this factor or more, quit.

```
resid_norm_jump_limit = 1d99
```

max_resid_jump_limit ¶

If residual max increases by this factor or more, quit.

```
max_resid_jump_limit = 1d6
```

max_iterations_for_jacobian ¶

EXPERIEMENTAL: not working at present. leave at 1. Jacobian is always created fresh for 1st iteration. If this param > 1, then will try to reuse jacobian. After use jacobian this many times, remake it. E.g., if = 2, then

will make a new jacobian for every other iteration. This is automatically = 1 immediately following a backup.

```
max_iterations_for_jacobian = 1
```

convergence_ignore_equL_residuals ¶

```
convergence_ignore_equL_residuals = .false.
```

trace_newton_damping ¶

Send newton damping data to screen.

```
trace_newton_damping = .false.
```

hydro_decsol_switch ¶

small_mtx_decsol ¶

large_mtx_decsol ¶

If current `nvar <= hydro_decsol_switch`, (recall `nvar = nvar_hydro + species`) then use `small_mtx_decsol` for current step, else use `large_mtx_decsol`.

Options for `small_mtx_decsol` are `'block_thomas_dble'` or `'bcyclic_dble'`.

Options for `large_mtx_decsol` are `'bcyclic_klu'`.

```
hydro_decsol_switch = 99999999
small_mtx_decsol = 'bcyclic_dble'
large_mtx_decsol = 'bcyclic_klu'
```

star_bcyclic_do_pivot ¶

Controls whether or not do pivoting in matrix solves in star bcyclic.

```
star_bcyclic_do_pivot = .true.
```

max_tries ¶

Max number newton iterations before give up.

```
max_tries = 25
```

max_tries1 ¶

Max tries on 1st model.

```
max_tries1 = 250
```

max_tries_for_retry ¶

Normal number of retries.

```
max_tries_for_retry = 25
```

max_tries_after_5_retries ¶

Increase number of tries after 5 failed ones.

```
max_tries_after_5_retries = 35
```

max_tries_after_10_retries ¶

Increase number of tries after 10 failed ones.

```
max_tries_after_10_retries = 50
```

max_tries_after_20_retries ¶

Increase number of tries after 20 failed ones.

```
max_tries_after_20_retries = 75
```

max_tries_after_backup ¶

Max tries after first backup.

```
max_tries_after_backup = 25
```

max_tries_after_backup2 ¶

Max tries after second backup.

```
max_tries_after_backup2 = 25
```

retry_limit ¶

Only use if > 0. In case the solver fails for some reason, it will retry with a smaller timestep. It does up to this many retries for the current step before doing a backup to the previous step.

```
retry_limit = 2
```

redo_limit ¶

Only use if > 0. Do up to this many redo's for the current step before doing a backup to the previous step.

```
redo_limit = 100
```

newton_itermin ¶

Use at least this many iterations in newton for hydro solve.

```
newton_itermin = 2
```

newton_itermin_until_reduce_min_corr_coeff ¶

Use at least this many iterations in newton before try using small `min_corr_coeff`

```
newton_itermin_until_reduce_min_corr_coeff = 8
```

newton_reduced_min_corr_coeff ¶

For use with `newton_itermin_for_reduce_min_corr_coeff`.

```
newton_reduced_min_corr_coeff = 0.1d0
```

tiny_corr_coeff_limit ¶

scale_correction_norm ¶

corr_param_factor ¶

scale_max_correction ¶

corr_coeff_limit ¶

tiny_corr_factor ¶

see star/private/star_newton for info about these

```
tiny_corr_coeff_limit = 5
scale_correction_norm = 0.1
```

```
corr_param_factor = 10
scale_max_correction = 1d99
corr_coeff_limit = 1d-2
tiny_corr_factor = 2
```

min_xa_hard_limit ¶

min_xa_hard_limit_for_highT ¶

If solver produces mass fraction < this limit, then reject the trial solution. Can optionally relax this limit at high T.

```
min_xa_hard_limit = -1d-5
min_xa_hard_limit_for_highT = -3d-5
```

logT_max_for_xa_hard_limit ¶

Use `min_xa_hard_limit` for center logT <= this.

```
logT_max_for_min_xa_hard_limit = 9.49d0
```

logT_min_for_xa_hard_limit_for_highT ¶

Use `min_xa_hard_limit_for_highT` for center logT >= this. Linear interpolate in logT for intermediate center temperatures.

```
logT_min_for_min_xa_hard_limit_for_highT = 9.51d0
```

sum_xa_hard_limit ¶

sum_xa_hard_limit_for_highT ¶

If solver produces any cell with abs(sum(xa)-1) > this limit, then reject the trial solution. Can optionally relax this limit at high T.

```
sum_xa_hard_limit = 5d-4
sum_xa_hard_limit_for_highT = 1d-3
```

logT_max_for_sum_xa_hard_limit ¶

Use `sum_xa_hard_limit` for center logT <= this.

```
logT_max_for_sum_xa_hard_limit = 9.49d0
```

logT_min_for_sum_xa_hard_limit_for_highT ¶

Use `sum_xa_hard_limit_for_highT` for center logT >= this. Linear interpolate in logT for intermediate center temperatures.

```
logT_min_for_sum_xa_hard_limit_for_highT = 9.51d0
```

do_newton_damping_for_neg_xa ¶

If true, uniformly reduce newton corrections if necessary to avoid neg abundances.

```
do_newton_damping_for_neg_xa = .true.
```

min_logT_for_quad ¶

```
min_logT_for_quad = 99
```

min_chem_eqn_scale ¶

```
min_chem_eqn_scale = 1d0
```

hydro_mtx_max_allowed_{abs}{dlogT | dlogRho | dlogPgas | logT | logRho | logPgas} ¶

Force retry with smaller timestep if hydro solves change T, Rho, or Pgas by too much or make them too large.

```
hydro_mtx_max_allowed_abs_dlogT = 99d0
hydro_mtx_max_allowed_abs_dlogE = 99d0
hydro_mtx_max_allowed_abs_dlogRho = 99d0
hydro_mtx_max_allowed_abs_dlogPgas = 99d0
min_logT_for_hydro_mtx_max_allowed = -1d99
hydro_mtx_max_allowed_logT = 99d0
hydro_mtx_max_allowed_logE = 99d0
hydro_mtx_max_allowed_logRho = 99d0
hydro_mtx_max_allowed_logPgas = 99d0
```

solver_clip_dlogT ¶

solver_clip_dlogRho ¶

solver_clip_dlogPgas ¶

solver_clip_dlogR ¶

Limit magnitude of relative changes per iteration in solver. Ignore if limit is <= 0.

```
solver_clip_dlogT = -1d0
solver_clip_dlogRho = -1d0
```

```
solver_clip_dlogPgas = -1d0
solver_clip_dlogR = -1d0
```

use_time_centering ¶

if true, then use time centered velocity and time weighted area.

```
use_time_centering = .false.
```

gold tolerances for newton solver

use_gold_tolerances ¶

gold_newton_iterations_limit ¶

gold_newton_iterations_hard_limit ¶

maxT_for_gold_tolerances ¶

max_eosPC_frac_for_gold_tolerances ¶

gold_tol_residual_norm1 ¶

gold_iter_for_resid_tol2 ¶

gold_tol_residual_norm2 ¶

gold_tol_max_residual2 ¶

gold_iter_for_resid_tol3 ¶

gold_tol_residual_norm3 ¶

gold_tol_max_residual3 ¶

```
use_gold_tolerances = .false.
```

```
maxT_for_gold_tolerances = 7e8
max_eosPC_frac_for_gold_tolerances = 1d-9
gold_tol_residual_norm1 = 1d-11
gold_tol_max_residual1 = 1d-9
gold_iter_for_resid_tol2 = 9
gold_tol_residual_norm2 = 1d-8
gold_tol_max_residual2 = 1d-6
gold_iter_for_resid_tol3 = 14
gold_tol_residual_norm3 = 1d-6
gold_tol_max_residual3 = 1d-4
```

```
gold_newton_iterations_limit = 14
gold_newton_iterations_hard_limit = -1
```

artificial viscosity

eps_visc_factor ¶

multiply the eps_visc term in energy equation by this factor.

```
eps_visc_factor = 1d0
```

dvdt_visc_factor ¶

multiply the dvdt_visc term in momentum equation by this factor.

```
dvdt_visc_factor = 1d0
```

use_artificial_viscosity ¶

artificial viscosity – only applies when using velocity variables

```
use_artificial_viscosity = .false.
```

artificial_viscosity_Q_shift ¶

```
 Qvisc = min(0d0, Qvisc + artificial_viscosity_Q_shift)
```

This serves to filter out use of artificial viscosity in low compression regions. e.g., try `artificial_viscosity_Q_shift = 1d34`.

```
artificial_viscosity_Q_shift = -1
```

post_shock_viscosity_decay_factor ¶

Exponential decrease in artificial viscosity inward from Mach 1 location. The value of Qvisc is multiplied by `exp(-dist_to_Mach1/Hq)` where Hq = this factor times the local radius and `dist_to_Mach1` is the distance at the start of the current step outward to the nearest Mach 1 location if using both pre and post shock decay, use the closer Mach1

```
post_shock_viscosity_decay_factor = -1
```

pre_shock_viscosity_decay_factor ¶

Exponential decrease in artificial viscosity outward from Mach 1 location. The value of Qvisc is multiplied by `exp(-dist_to_Mach1/Hq)` where Hq = this factor times the local radius and `dist_to_Mach1` is the distance at the start of the current step inward to the nearest Mach 1 location if using both pre and post shock decay, use the closer Mach1

```
pre_shock_viscosity_decay_factor = -1
```

shock_spread_quadratic ¶

the artificial viscosity coefficient includes a quadratic term that is proportional to `(shock_spread_quadratic * r)^2` where r is the local radius.

```
shock_spread_quadratic = 1d-3
```

shock_spread_linear ¶

the artificial viscosity coefficient includes a linear term that is proportional to `(shock_spread_linear * r * cs)` where cs is the local sound speed and r is the local radius.

```
shock_spread_linear = 0
```

art_visc_full_on_logRho_ge_this ¶

art_visc_full_off_logRho_le_this ¶

```
art_visc_full_on_logRho_ge_this = -99
art_visc_full_off_logRho_le_this = -99
```

split mixing ¶

split_mixing_choice ¶

- 0 = no split = mixing coupled to burn and structure.
- -1 = mix for full dt before burn+struct
- -2 = mix for full dt after burn+struct
- -3 = mix for dt/2 before and dt/2 after

```
split_mixing_choice = 0
```

reset_mixing_info_before_final_mix ¶

Relevant for `split_mixing_choice` options -2 and -3.

```
reset_mixing_info_before_final_mix = .true.
```

op_split_mix_atol ¶

op_split_mix_rtol ¶

Abolute and relative error tolerances.

```
op_split_mix_atol = 1d-5
op_split_mix_rtol = 1d-6
```

controls related to split mixing for timesteps

max_fixup_for_mix_limit ¶

If `split_mix_fixup` is bigger than this, reduce the next timestep.

```
max_fixup_for_mix_limit = 1d-3
```

max_fixup_for_mix_hard_limit ¶

If `split_mix_fixup` is bigger than this, retry.

```
max_fixup_for_mix_hard_limit = 1d99
```

op_split_mix_trace ¶

```
op_split_mix_trace = .false.
```

op_split_burn ¶

```
op_split_burn = .false.
```

timestep controls ¶

The terminal output during evolution includes a short string for the `dt_limit`. This is to give you some indication of what is limiting the time steps. Here's a dictionary mapping those terminal strings to the corresponding control parameters. (There is a similar table in `mesa/binary/defaults/binary_controls.defaults`.)

```
      terminal output       related parameter
     'avg lgE resid'        limit_for_avg_lgE_residual
     'CpT_absMdot_div_L'    CpT_absMdot_div_L_limit
     'Lnuc'                 delta_lgL_nuc_limit
     'Lnuc_cat'             delta_lgL_nuc_cat_limit
     'Lnuc_H'               delta_lgL_H_limit
```

```
      'Lnuc_He'             delta_lgL_He_limit
      'Lnuc_photo'          delta_lgL_photo_limit
      'Lnuc_z'              delta_lgL_z_limit
      'bad_X_sum'           (solver found bad mass sum)
      'dH'                  dH_limit
      'dH/H'                dH_div_H_limit
      'dHe'                 dHe_limit
      'dHe/He'              dHe_div_He_limit
      'dHe3'                dHe3_limit
      'dHe3/He3'            dHe3_div_He3_limit
      'dL/L'                dL_div_L_limit
      'dX'                  dX_limit
      'dX/X'                dX_div_X_limit
      'dX_nuc_drop'         dX_nuc_drop_limit
      'd_delR_grow'         d_deltaR_grow_limit
      'd_delR_shrink'       d_deltaR_shrink_limit
      'delta Ye'            delta_Ye_limit
      'delta mdot'          delta_mdot_limit
      'delta total J'       delta_lg_total_J_limit
      'delta_HR'            delta_HR_limit
      'delta_mstar'         delta_lg_star_mass_limit
      'diff iters'          diffusion_iters_limit
      'diff steps'          diffusion_steps_limit
      'min_dr_div_cs'       dt_div_min_dr_div_cs_limit
      'dt_acoustic'         dt_div_dt_acoustic_limit
      'dt_collapse'         dt_div_dt_cell_collapse_limit
      'dt_dynamic'          dt_div_dt_dynamic_limit
      'dt_mass_loss'        dt_div_dt_mass_loss_limit
      'dt_thermal'          dt_div_dt_thermal_limit
      'eps_nuc_cntr'        delta_log_eps_nuc_cntr_limit
      'error enrg'          limit_for_rel_error_in_energy_conservation
      'error rate'          limit_for_log_rel_rate_in_energy_conservat
      'highT del Ye'        delta_Ye_highT_limit
      'hold'                (recent backup, so no increase in dt)
      'lgL'                 delta_lgL_limit
      'lgL_phot'            delta_lgL_phot_limit
      'lgP'                 delta_lgP_limit
      'lgP_cntr'            delta_lgP_cntr_limit
      'lgR'                 delta_lgR_limit
      'lgRho'               delta_lgRho_limit
      'lgRho_cntr'          delta_lgRho_cntr_limit
      'lgRho_max'           delta_lgRho_max_limit
      'lgT'                 delta_lgT_limit
      'lgT_cntr'            delta_lgT_cntr_limit
      'lgT_max'             delta_lgT_max_limit
      'lgTeff'              delta_lgTeff_limit
      'lg_XC_cntr'          delta_lg_XC_cntr_limit
      'lg_XH_cntr'          delta_lg_XH_cntr_limit
      'lg_XHe_cntr'         delta_lg_XHe_cntr_limit
      'lg_XNe_cntr'         delta_lg_XNe_cntr_limit
      'lg_XO_cntr'          delta_lg_XO_cntr_limit
      'lg_XSi_cntr'         delta_lg_XSi_cntr_limit
      'log_eps_nuc'         delta_log_eps_nuc_limit
      'max E resid'         limit_for_max_E_residual
      'max lgE resid'       limit_for_max_abs_lgE_residual
      'max_dt'              max_years_for_timestep
      'max dt change'       max_timestep_factor
      'min dt change'       min_timestep_factor
```

```
      'neg_mass_frac'         (solver found neg mass frac)
      'newton iters'          newton_iterations_limit
      'rotation steps'        rotation_steps_limit
      'v/v_crit'              v_div_v_crit_limit
      'varcontrol'            varcontrol_target
      'b_****'                see binary/defaults/binary_controls.defaul
```

max_timestep ¶

In seconds. `max_timestep <= 0` means no upper limit.

```
max_timestep = 0
```

max_years_for_timestep ¶

`max_years_for_timestep <= 0` means no upper limit. Note: `max_timestep` is the control that is used by most of the code. `max_years_for_timestep` is just provided as a convenience. At the start of each step, the evolve routine checks to see if `max_years_for_timestep > 0`, and if so, it sets `max_timestep = max_years_for_timestep*secyer`.

```
max_years_for_timestep = 0
```

max_timestep_hi_T_limit ¶

If `max T >= this`, then switch to `hi_T_max_years_for_timestep`. Ignore if <= 0.

```
max_timestep_hi_T_limit = -1
```

hi_T_max_years_for_timestep ¶

Max years for timestep if `max_timestep_hi_T_limit` is active.

```
hi_T_max_years_for_timestep = 0
```

min_timestep_factor ¶

Lower limit for ratio of new timestep to previous timestep. i.e., allow dt to get smaller by no more than this factor – 0 means no limit.

```
min_timestep_factor = 0.8d0
```

force_timestep_min ¶

In seconds. `force_timestep_min <= 0` means no forced lower limit.

```
force_timestep_min = 0
```

force_timestep_min_years ¶

`force_timestep_min_years <= 0` means no forced lower limit. Note: `force_timestep_min` is the control that is used by most of the code. `force_timestep_min_years` is just provided as a convenience. At the start of each step, the evolve routine checks if `force_timestep_min_years > 0`, and if so, it sets `force_timestep_min = force_timestep_min_years*secyer`.

```
force_timestep_min_years = 0
```

force_timestep_min_factor ¶

If dt is < `force_timestep_min`, then replace dt by `min(dt*force_timestep_min_factor, force_timestep_min)`

```
force_timestep_min_factor = 2d0
```

max_timestep_factor ¶

Upper limit for ratio of new timestep to previous timestep. i.e., allow dt to get larger by no more than this factor – 0 means no limit.

```
max_timestep_factor = 1.2d0
```

timestep_factor_for_retries ¶

Before retry, decrease dt by this.

```
timestep_factor_for_retries = 0.5d0
```

timestep_factor_for_backups ¶

Before backup, decrease dt by this (or more if multiple backups in a row).

```
timestep_factor_for_backups = 0.5d0
```

backup_hold ¶

No increases in timestep for `backup_hold` steps after a backup.

```
backup_hold = 2
```

retry_hold ¶

No increases in timestep for `retry_hold` steps after a retry.

```
retry_hold = 1
```

neg_mass_fraction_hold ¶

No increases in timestep for `neg_mass_fraction_hold` steps after a retry or backup caused by a negative mass fraction.

```
neg_mass_fraction_hold = 2
```

timestep_dt_factor = 0.9 ¶

dt reduction factor exceed timestep limits.

```
timestep_dt_factor = 0.9d0
```

dt_limit_ratio_target ¶

Aim for this ratio on dt limited timesteps.

```
dt_limit_ratio_target = 1d0
```

use_dt_low_pass_controller ¶

Enable low pass filter for smoother timestep variations.

```
use_dt_low_pass_controller = .true.
```

varcontrol_target ¶

This is the target value for relative variation in the structure from one model to the next. The default timestep adjustment is to increase or reduce the timestep depending on whether the actual variation was smaller or greater than this value.

```
varcontrol_target = 1d-4
```

varcontrol_dt_limit_ratio_hard_max ¶

`varcontrol_dt_limit_ratio` is the actual varcontrol value divided by the target. if that ratio exceeds this limit, then retry with a smaller timestep. this let's you prevent large changes from happening in a single step.

```
varcontrol_dt_limit_ratio_hard_max = 1d99
```

relax_hard_limits_after_backup ¶

If true, then don't enforce hard limits immediately after a backup.

```
relax_hard_limits_after_backup = .true.
```

relax_hard_limits_after_retry ¶

If true, then don't enforce hard limits immediately after a retry.

```
relax_hard_limits_after_retry = .true.
```

limits based on iterations required by various solvers

newton_iterations_limit ¶

If newton solve uses more `newton_iterations` than this, reduce the next timestep.

```
newton_iterations_limit = 7
```

newton_iterations_hard_limit ¶

If uses more iterations than this, retry.

```
newton_iterations_hard_limit = -1
```

rotation_steps_limit ¶

If rotation solver uses more steps than this, reduce the next timestep.

```
rotation_steps_limit = 500
```

rotation_steps_hard_limit ¶

If rotation solver uses more steps than this, retry.

```
rotation_steps_hard_limit = 700
```

diffusion_steps_limit ¶

If diffusion solver uses more steps than this, reduce the next timestep.

```
diffusion_steps_limit = 500
```

diffusion_steps_hard_limit ¶

If diffusion solver uses more steps than this, retry.

```
diffusion_steps_hard_limit = 700
```

diffusion_iters_limit ¶

If use a total number of iters > this, reduce the next timestep.

```
diffusion_iters_limit = 600
```

diffusion_iters_hard_limit ¶

If use a total number of iters > this, retry.

```
diffusion_iters_hard_limit = 800
```

limits based on max decrease in mass fraction at any location in star

dX_mix_dist_limit ¶

Option to ignore decreases in abundance in non-mixed cells near mixing boundaries. Ignore abundance changes if nearest mixing boundary is closer than this in Msun units. This applies to `dH`, `dH_div_H`, `dHe`, `dHe_d_He`, `dX`, and `dX_div_X` limits.

```
dX_mix_dist_limit = 1d-4
```

Limit on magnitude of decrease in any cell hydrogen abundance during a single timestep. dH here is `abs(xa(h1,k) - xa_old(h1,k))` for any cell k. Considers all cells except where have convective mixing.

dH_limit_min_H ¶

dH limits only apply where xa(h1,k) >= this limit.

```
dH_limit_min_H = 1d99
```

dH_limit ¶

If max dH is greater than this, reduce the next timestep by `dH_limit/max_dH`.

```
dH_limit = 1d99
```

dH_hard_limit ¶

If max dH is greater than this, retry with smaller timestep.

```
dH_hard_limit = 1d99
```

dH_decreases_only ¶

If true, then only consider decreases in abundance.

```
dH_decreases_only = .true.
```

Limit on magnitude of relative decrease in any cell hydrogen abundance. `dH_div_H` here is `abs(xa(h1,k) - xa_old(h1,k))/xa(h1,k)` considers all cells except where have convective mixing. `dH_decreases_only` applies to `dH_div_H` also.

dH_div_H_limit_min_H ¶

`dH_div_H` limits only apply where xa(h1,k) >= this limit.

```
dH_div_H_limit_min_H = 1d-3
```

dH_div_H_limit ¶

If max `dH_div_H` is greater than this, reduce the next timestep by `dH_limit/max_dH`.

```
dH_div_H_limit = 0.5d0
```

dH_div_H_hard_limit ¶

If max `dH_div_H` is greater than this, retry with smaller timestep.

```
dH_div_H_hard_limit = 1d99
```

Limit on magnitude of decrease in any cell helium abundance during a single timestep. dHe here is `abs(xa(he4,k) - xa_old(he4,k))` for any cell k. Considers all cells except where have convective mixing.

dHe_limit_min_He ¶

dHe limits only apply where xa(he4,k) >= this limit.

```
dHe_limit_min_He = 1d99
```

dHe_limit = 1d99 ¶

If max dHe is greater than this, reduce the next timestep by `dHe_limit/max_dHe`.

```
dHe_limit = 1d99
```

dHe_hard_limit ¶

If max dHe is greater than this, retry with smaller timestep.

```
dHe_hard_limit = 1d99
```

dHe_decreases_only ¶

If true, then only consider decreases in abundance. `dHe_decreases_only` applies to `dHe_div_He` also.

```
dHe_decreases_only = .true.
```

Limit on magnitude of relative decrease in any cell helium abundance. `dHe_div_He` here is `abs(xa(he4,k) - xa_old(he4,k))/xa(he4,k)`. Considers all cells except where have convective mixing.

dHe_div_He_limit_min_He ¶

`dHe_div_He` limits only apply where xa(he4,k) >= this limit.

```
dHe_div_He_limit_min_He = 1d-3
```

dHe_div_He_limit ¶

If max `dHe_div_He` is greater than this, reduce the next timestep by `dHe_limit/max_dHe`.

```
dHe_div_He_limit = 0.5d0
```

dHe_div_He_hard_limit ¶

If max `dHe_div_He` is greater than this, retry with smaller timestep.

```
dHe_div_He_hard_limit = 1d99
```

Limit on magnitude of decrease in any cell helium abundance during a single timestep. dHe3 here is `abs(xa(he4,k) - xa_old(he3,k))` for any cell k. Considers all cells except where have convective mixing.

dHe3_limit_min_He3 ¶

dHe3 limits only apply where xa(he3,k) >= this limit.

```
dHe3_limit_min_He3 = 1d99
```

dHe3_limit ¶

If max dHe3 is greater than this, reduce the next timestep by `dHe3_limit/max_dHe3`.

```
dHe3_limit = 1d99
```

dHe3_hard_limit ¶

If max dHe3 is greater than this, retry with smaller timestep.

```
dHe3_hard_limit = 1d99
```

dHe3_decreases_only ¶

If true, then only consider decreases in abundance. `dHe3_decreases_only` applies to `dHe3_div_He3` also.

```
dHe3_decreases_only = .true.
```

Limit on magnitude of relative decrease in any cell helium abundance. `dHe3_div_He3` here is `abs(xa(he3,k) - xa_old(he3,k))/xa(he3,k)`. Considers all cells except where have convective mixing.

dHe3_div_He3_limit_min_He3 ¶

`dHe3_div_He3` limits only apply where xa(he3,k) >= this limit.

```
dHe3_div_He3_limit_min_He3 = 1d99
```

dHe3_div_He3_limit ¶

if max `dHe3_div_He3` is greater than this, reduce the next timestep by `dHe3_limit/max_dHe3`.

```
dHe3_div_He3_limit = 1d99
```

dHe3_div_He3_hard_limit ¶

If max `dHe3_div_He3` is greater than this, retry with smaller timestep.

```
dHe3_div_He3_hard_limit = 1d99
```

Limit on magnitude of decrease in any cell nonH, nonHe abundance. dX here is `abs(xa(j,k) - xa_old(j,k))` for any cell k and any species j other except hydrogen or helium. Considers all cells except where have convective mixing.

dX_limit_min_X ¶

dX limits only apply where xa(j,k) >= this limit.

```
dX_limit_min_X = 1d99
```

dX_limit ¶

If max dX is greater than this, reduce the next timestep by `dX_limit/max_dX`.

```
dX_limit = 1d99
```

dX_hard_limit ¶

If max dX is greater than this, retry with smaller timestep.

```
dX_hard_limit = 1d99
```

dX_decreases_only ¶

If true, then only consider decreases in abundance. `dX_decreases_only` applies to `dX_div_X` also.

```
dX_decreases_only = .true.
```

Limit on magnitude of relative decrease in any cell nonH, nonHe abundance. `dX_div_X` here is abs(xa(j,k) - xa_old(j,k))/xa(j,k) for any cell k and any species j other except hydrogen or helium. Considers all cells except where have convective mixing.

dX_div_X_limit_min_X ¶

`dX_div_X` limits only apply where xa(j,k) >= this limit.

```
dX_div_X_limit_min_X = 1d99
```

dX_div_X_limit ¶

If max `dX_div_X` is greater than this, reduce the next timestep by `dX_limit/max_dX`.

```
dX_div_X_limit = 1d99
```

dX_div_X_hard_limit ¶

If max dX_div_X is greater than this, retry with smaller timestep.

```
dX_div_X_hard_limit = 1d99
```

Limits on max drop in abundance mass fraction from burning with possible mixing inflow. This considers both nuclear reactions and offsetting effect of mixing inflow.

dX_nuc_drop_min_X_limit ¶

dX_nuc_drop_limit only for X > dX_nuc_drop_min_X_limit.

```
dX_nuc_drop_min_X_limit = 1d-4
```

dX_nuc_drop_max_A_limit ¶

dX_nuc_drop_limit only for species with A <= dX_nuc_drop_max_A_limit.

```
dX_nuc_drop_max_A_limit = 52
```

dX_nuc_drop_limit_at_high_T ¶

Negative means use value for dX_nuc_drop_limit, else use this limit when center logT > 9.45.

```
dX_nuc_drop_limit_at_high_T = -1
```

dX_nuc_drop_limit ¶

If max dX_nuc_drop is greater than dX_nuc_drop_limit, reduce the next timestep by dX_nuc_drop_limit/max_dX_nuc_drop.

```
dX_nuc_drop_limit = 5d-2
```

dX_nuc_drop_hard_limit ¶

If max dX_nuc_drop is greater than dX_nuc_drop_hard_limit, retry with smaller timestep.

```
dX_nuc_drop_hard_limit = 1d99
```

dX_nuc_drop_min_yrs_for_dt ¶

Don’t let dX_nuc_drop change dt to smaller than this.

```
dX_nuc_drop_min_yrs_for_dt = 1d-9
```

limits based on relative changes in variables L, P, Rho, T, R, eps_nuc ¶

limit on magnitude of relative change in L at any grid point

```
dL_div_L = abs(L(k) - L_old(k))/L(k)
```

dL_div_L_limit ¶

If max abs `dL_div_L` is greater than this, reduce the next timestep.

```
dL_div_L_limit = -1
```

dL_div_L_hard_limit ¶

If max abs `dL_div_L` is greater than this, retry with smaller timestep.

```
dL_div_L_hard_limit = -1
```

dL_div_L_limit_min_L ¶

In Lsun units. `dL_div_L` limits only apply where `L(k) >= Lsun*dL_limit_min_L`

```
dL_div_L_limit_min_L = 1d99
```

delta_lgP_limit ¶

Limit for magnitude of max change in log10 total pressure in any cell.

```
delta_lgP_limit = 1
```

delta_lgP_hard_limit ¶

If max `delta_lgP` is greater than `delta_lgP_hard_limit`, retry with smaller timestep.

```
delta_lgP_hard_limit = -1
```

delta_lgP_limit_min_lgP ¶

`delta_lgP_limit` limits only apply where `log10_P(k) >= delta_lgP_limit_min_lgP`

```
delta_lgP_limit_min_lgP = 1d99
```

delta_lgRho_limit ¶

Limit for magnitude of max change in log10 density in any cell.

```
delta_lgRho_limit = 1
```

delta_lgRho_hard_limit = -1 ¶

If max `delta_lgRho` is greater than `delta_lgRho_hard_limit`, retry with smaller timestep.

```
delta_lgRho_hard_limit = -1
```

delta_lgRho_limit_min_lgRho ¶

`delta_lgRho_limit` limits only apply where `log10_Rho(k) >= delta_lgRho_limit_min_lgRho`.

```
delta_lgRho_limit_min_lgRho = 1d99
```

delta_lgT_limit ¶

Limit for magnitude of max change in log10 temperature in any cell.

```
delta_lgT_limit = 0.5d0
```

delta_lgT_hard_limit ¶

If max `delta_lgT` is greater than `delta_lgT_hard_limit`, retry with smaller timestep.

```
delta_lgT_hard_limit = -1
```

delta_lgT_limit_min_lgT ¶

`delta_lgT_limit` limits only apply where `log10_T(k) >= delta_lgT_limit_min_lgT`.

```
delta_lgT_limit_min_lgT = 1d99
```

delta_lgE_limit ¶

Limit for magnitude of max change in log10 internal energy in any cell.

```
delta_lgE_limit = 0.1d0
```

delta_lgE_hard_limit ¶

If max `delta_lgE` is greater than `delta_lgE_hard_limit`, retry with smaller timestep.

```
delta_lgE_hard_limit = -1
```

delta_lgE_limit_min_lgE ¶

`delta_lgE_limit` limits only apply where `log10(E(k)) >= delta_lgE_limit_min_lgE`.

```
delta_lgE_limit_min_lgE = 1d99
```

delta_lgR_limit ¶

Limit for magnitude of max change in log10 radius at any cell boundary.

```
delta_lgR_limit = 0.5d0
```

delta_lgR_hard_limit ¶

If max `delta_lgR` is greater than `delta_lgR_hard_limit`, retry with smaller timestep.

```
delta_lgR_hard_limit = -1
```

delta_lgR_limit_min_lgR ¶

`delta_lgR_limit` limits only apply where `log10_R(k) >= delta_lgR_limit_min_lgR`.

```
delta_lgR_limit_min_lgR = 1d99
```

delta_Ye_limit ¶

Limit for magnitude of max change in Ye in any cell.

```
delta_Ye_limit = 1
```

delta_Ye_hard_limit ¶

If max `delta_Ye` is greater than `delta_Ye_hard_limit`, retry with smaller timestep.

```
delta_Ye_hard_limit = -1
```

delta_Ye_highT_limit ¶

Limit for magnitude of max change in Ye in high T cells.

```
delta_Ye_highT_limit = 99
```

Limit testing for max `delta_ye` to cells with `T >= minT_for_highT_Ye_limit` If this high T max `delta_Ye` is greater than `delta_Ye_highT_limit`, reduce the next timestep by `delta_Ye_highT_limit/max_delta_Ye`.

```
delta_Ye_highT_hard_limit = -1
```

minT_for_highT_Ye_limit ¶

Limit testing for max `delta_ye` to cells with `T >= minT_for_highT_Ye_limit`. If this high T max `delta_Ye` is greater than `delta_Ye_highT_limit`, retry with smaller timestep.

```
minT_for_highT_Ye_limit = 7d9
```

delta_log_eps_nuc_limit ¶

Limit for magnitude of max change in log10 `eps_nuc` in any cell. Only applies to increases in non-convective zones.

```
delta_log_eps_nuc_limit = -1
```

delta_log_eps_nuc_hard_limit ¶

If max `delta_log_eps_nuc` is greater than `delta_log_eps_nuc_hard_limit`, retry with smaller timestep.

```
delta_log_eps_nuc_hard_limit = -1
```

d_deltaR_shrink_limit ¶

Limit for relative decrease in radial thickness of any zone.

```
d_deltaR_shrink_limit = -1
```

d_deltaR_shrink_hard_limit ¶

If max `d_deltaR_shrink` is greater than `d_deltaR_shrink_hard_limit`, retry with smaller timestep.

```
d_deltaR_shrink_hard_limit = -1
```

d_deltaR_grow_limit ¶

Limit for relative increase in radial thickness of any zone.

```
d_deltaR_grow_limit = -1
```

d_deltaR_grow_hard_limit ¶

If max `d_deltaR_grow` is greater than `d_deltaR_grow_hard_limit`, retry with smaller timestep.

```
d_deltaR_grow_hard_limit = -1
```

limits based on integrated power at each point for each category of nuclear reaction ¶

`lgL_nuc_cat` = nuclear reaction energy release for a particular category of reaction (Lsun units). Energy release here excludes neutrinos.

delta_lgL_nuc_cat_limit ¶

Limit for magnitude of change in `lgL_nuc` for category.

```
delta_lgL_nuc_cat_limit = -1
```

delta_lgL_nuc_cat_hard_limit ¶

If max delta is greater than `delta_lgL_nuc_cat_hard_limit`, retry with smaller timestep.

```
delta_lgL_nuc_cat_hard_limit = -1
```

lgL_nuc_cat_burn_min ¶

Ignore changes in `lgL_nuc` for category if value is less than this.

```
lgL_nuc_cat_burn_min = -1
```

lgL_nuc_mix_dist_limit ¶

Ignore if nearest boundary is closer than this. Ignore changes in lgL in cells near mixing boundaries.

```
lgL_nuc_mix_dist_limit = 1d-6
```

check_deltalgL_{burning_category} ¶

Flags determining which reaction categories are considered.

```
check_delta_lgL_pp = .true.
check_delta_lgL_cno = .true.
check_delta_lgL_3alf = .true.
check_delta_lgL_burn_c = .true.
check_delta_lgL_burn_n = .true.
check_delta_lgL_burn_o = .true.
check_delta_lgL_burn_ne = .true.
check_delta_lgL_burn_na = .true.
check_delta_lgL_burn_mg = .true.
check_delta_lgL_burn_si = .true.
check_delta_lgL_burn_s = .true.
check_delta_lgL_burn_ar = .true.
check_delta_lgL_burn_ca = .true.
check_delta_lgL_burn_ti = .true.
check_delta_lgL_burn_cr = .true.
check_delta_lgL_burn_fe = .true.
```

c12 + c12, c12 + o16, and o16 + o16

```
check_delta_lgL_cc = .true.
check_delta_lgL_co = .true.
check_delta_lgL_oo = .true.
```

`L_H_burn` = integrated power at surface from PP and CNO (in Lsun units)

values for `lgL_H` are `log10(max(1, L_H_burn))`

delta_lgL_H_limit ¶

limit for magnitude of change in `lgL_H`

```
delta_lgL_H_limit = -1
```

delta_lgL_H_hard_limit ¶

if max delta is greater than `delta_lgL_H_hard_limit`, retry with smaller timestep

```
delta_lgL_H_hard_limit = -1
```

lgL_H_burn_min ¶

ignore changes in `lgL_H` if value is less than this

```
lgL_H_burn_min = 1.5d0
```

lgL_H_drop_factor ¶

when `L_H` is dropping, multiply limits by this factor

```
lgL_H_drop_factor = 1
```

lgL_H_burn_relative_limit ¶

ignore changes in `lgL_H` if `max(lgL_He,lgL_z) - lgL_H > this`

```
lgL_H_burn_relative_limit = 3
```

`L_He_burn` = integrated power at surface from triple alpha (in Lsun units)

values for `lgL_He` are `log10(max(1, L_He_burn))`

delta_lgL_He_limit ¶

Limit for magnitude of change in lgL_He.

```
delta_lgL_He_limit = 0.025d0
```

delta_lgL_He_hard_limit ¶

If max delta is greater than `delta_lgL_He_hard_limit`, retry with smaller timestep.

```
delta_lgL_He_hard_limit = -1
```

lgL_He_burn_min ¶

Ignore changes in `lgL_He` if value is less than this.

```
lgL_He_burn_min = 2.5d0
```

lgL_He_drop_factor ¶

When `L_He` is dropping, multiply limits by this factor.

```
lgL_He_drop_factor = 1
```

lgL_He_burn_relative_limit ¶

Ignore changes in `lgL_He` if `max(lgL_H,lgL_z) - lgL_He > this`.

```
lgL_He_burn_relative_limit = 3
```

`L_z_burn` = integrated power at surface from nuclear burning other than H, He, or C (in Lsun units) excluding photodistintegrations

values for `lgL_z` are `log10(max(1, L_z_burn))`

delta_lgL_z_limit ¶

Limit for magnitude of change in `lgL_z`.

```
delta_lgL_z_limit = -1
```

delta_lgL_z_hard_limit ¶

If max delta is greater than `delta_lgL_z_hard_limit`, retry with smaller timestep.

```
delta_lgL_z_hard_limit = -1
```

lgL_z_burn_min ¶

Ignore changes in `lgL_z` if value is less than this.

```
lgL_z_burn_min = 2.5d0
```

lgL_z_drop_factor ¶

When `L_z` is dropping, multiply limits by this factor.

```
lgL_z_drop_factor = 1
```

lgL_z_burn_relative_limit ¶

Ignore changes in `lgL_z` if `max(lgL_H,lgL_He) - lgL_z > this`.

```
lgL_z_burn_relative_limit = 3
```

`L_photo_burn` = magnitude of integrated power at surface from photodistintegrations

values for `lgL_photo` are based on `L_by_category(iphoto)`

delta_lgL_photo_limit ¶

Limit for magnitude of change in `lgL_photo`.

```
delta_lgL_photo_limit = -1
```

delta_lgL_photo_hard_limit ¶

If max delta is greater than `delta_lgL_photo_hard_limit`, retry with smaller timestep.

```
delta_lgL_photo_hard_limit = -1
```

lgL_photo_burn_min ¶

Ignore changes in `lgL_photo` if value is less than this.

```
lgL_photo_burn_min = 2.5d0
```

lgL_photo_drop_factor ¶

When `L_photo` is dropping, multiply limits by this factor.

```
lgL_photo_drop_factor = 1
```

limits based on total integrated power at surface for all nuclear reactions ¶

excluding photodistintegrations

`L_nuc` = nuclear reaction total energy release for all nuclear reactions (Lsun units)

delta_lgL_nuc_limit ¶

limit for magnitude of change in `lgL_nuc`

```
delta_lgL_nuc_limit = -1
```

delta_lgL_nuc_hard_limit ¶

if max delta is greater than `delta_lgL_nuc_hard_limit`, retry with smaller timestep

```
delta_lgL_nuc_hard_limit = -1
```

lgL_nuc_burn_min ¶

ignore changes in `lgL_nuc` if value is less than this

```
lgL_nuc_burn_min = 0.5d0
```

lgL_nuc_drop_factor ¶

When L_nuc is dropping, multiply limits by this factor.

```
lgL_nuc_drop_factor = 1
```

limits based on changes at photosphere

delta_lgTeff_limit ¶

delta_lgTeff_hard_limit ¶

Limit for magnitude of max change in log10 temperature at photosphere.

```
delta_lgTeff_limit = 0.01d0
delta_lgTeff_hard_limit = -1
```

delta_lgL_limit_L_min ¶

delta_lgL_limit ¶

delta_lgL_hard_limit ¶

Limit for magnitude of change in log10(L/Lsun). Only apply this limit when L >= delta_lgL_limit_L_min (in Lsun units).

```
delta_lgL_limit_L_min = -100
delta_lgL_limit = 0.1d0
delta_lgL_hard_limit = -1
```

delta_lgL_phot_limit_L_min ¶

delta_lgL_phot_limit ¶

delta_lgL_phot_hard_limit ¶

Limit for magnitude of change in log10(L_phot/Lsun). Only apply this limit when L_phot >= delta_lgL_phot_limit_L_min (in Lsun units).

```
delta_lgL_phot_limit_L_min = -100
delta_lgL_phot_limit = 0.1d0
delta_lgL_phot_hard_limit = -1
```

v_div_v_crit_limit ¶

v_div_v_crit_hard_limit ¶

Limit surface rotational velocity div critical velocity (`v_div_v_crit_avg_surf`).

```
v_div_v_crit_limit = -1
v_div_v_crit_hard_limit = -1
```

dt_div_dt_thermal_limit ¶

dt_div_dt_thermal_hard_limit ¶

limit for dt compared to thermal timescale (negative means no limit)

```
 dt_thermal = (3/4)*G*M^2/(R*L); Kelvin-Helmholtz time
```

```
dt_div_dt_thermal_limit = -1
dt_div_dt_thermal_hard_limit = -1
```

dt_div_dt_dynamic_limit ¶

dt_div_dt_dynamic_hard_limit ¶

limit for dt compared to dynamic timescale (negative means no limit)

```
 dt_dynamic =  2*Pi*sqrt(R^3/(G*M))
```

```
dt_div_dt_dynamic_limit = -1
dt_div_dt_dynamic_hard_limit = -1
```

dt_div_dt_acoustic_limit ¶

dt_div_dt_acoustic_hard_limit ¶

limit for dt compared to dt_acoustic (negative means no limit)

dt_acoustic = time for sound from center to photosphere = sum over shells of local sound crossing time dr/csound.

```
dt_div_dt_acoustic_limit = -1
dt_div_dt_acoustic_hard_limit = -1
```

dt_div_dt_mass_loss_limit ¶

dt_div_dt_mass_loss_hard_limit ¶

limit for dt compared to mass loss timescale (negative means no limit)

```
dt_mass_loss = -M/Mdot; only applies when Mdot < 0
```

```
dt_div_dt_mass_loss_limit = -1
dt_div_dt_mass_loss_hard_limit = -1
```

dt_div_min_dr_div_cs_limit ¶

dt_div_min_dr_div_cs_hard_limit ¶

limit for dt compared to explicit solver timescale (negative means no limit)

```
min_dr_div_cs = min over all cells of dr/csound (seconds)
```

```
dt_div_min_dr_div_cs_limit = -1
dt_div_min_dr_div_cs_hard_limit = -1
```

min_k_for_dt_div_min_dr_div_cs_limit ¶

min_q_for_dt_div_min_dr_div_cs_limit ¶

max_q_for_dt_div_min_dr_div_cs_limit ¶

```
min_k_for_dt_div_min_dr_div_cs_limit = 20
min_q_for_dt_div_min_dr_div_cs_limit = 0.005d0
max_q_for_dt_div_min_dr_div_cs_limit = 0.995d0
```

min_abs_du_div_cs_for_dt_div_min_dr_div_cs_limit ¶

only use `dt_div_min_dr_div_cs_limit` at cells where `abs_du_div_cs` > this limit. allow focus on regions near shock face.

```
min_abs_du_div_cs_for_dt_div_min_dr_div_cs_limit = 0.01d0
```

dt_div_dt_cell_collapse_limit ¶

dt_div_dt_cell_collapse_hard_limit ¶

limit for dt compared to cell_collapse timescale (negative means no limit)

```
dt_cell_collapse = min over shells k that have v(k+1) > v(k) of
(r(k)-r(k+1))/(v(k+1)-v(k)), the time for the cell to collapse
to zero thickness at current velocities.
```

```
dt_div_dt_cell_collapse_limit = -1
dt_div_dt_cell_collapse_hard_limit = -1
```

limits based on changes in location on HR diagram

delta_HR_ds_L ¶

delta_HR_ds_Teff ¶

```
dlgL = log10(L/L_prev)
dlgTeff = log10(Teff/Teff_prev)
```

```
delta_HR_ds_L = 1
delta_HR_ds_Teff = 1
```

delta_HR_limit ¶

delta_HR_hard_limit ¶

limit for dHR (negative means no limit)

```
dHR = sqrt((delta_HR_ds_L*dlgL)**2 + (delta_HR_ds_Teff*dlgTeff)**2)
```

```
delta_HR_limit = -1
delta_HR_hard_limit = -1
```

limits based on change in max temperature or density

delta_lgT_max_limit ¶

delta_lgT_max_hard_limit ¶

limit for magnitude of change in log10 max temperature

```
delta_lgT_max_limit = -1
delta_lgT_max_hard_limit = -1
```

delta_lgRho_max_limit ¶

delta_lgRho_max_hard_limit ¶

limit for magnitude of change in log10 max density

```
delta_lgRho_max_limit = -1
delta_lgRho_max_hard_limit = -1
```

limits based on changes at center

delta_lgT_cntr_limit ¶

delta_lgT_cntr_hard_limit ¶

limit for magnitude of change in log10 temperature at center

```
delta_lgT_cntr_limit = 0.01d0
delta_lgT_cntr_hard_limit = -1
```

delta_lgP_cntr_limit ¶

delta_lgP_cntr_hard_limit ¶

limit for magnitude of change in log10 pressure at center

```
delta_lgP_cntr_limit = -1
delta_lgP_cntr_hard_limit = -1
```

delta_lgRho_cntr_limit ¶

delta_lgRho_cntr_hard_limit ¶

limit for magnitude of change in log10 density at center

```
delta_lgRho_cntr_limit = 0.05d0
delta_lgRho_cntr_hard_limit = -1
```

delta_log_eps_nuc_cntr_limit ¶

delta_log_eps_nuc_cntr_hard_limit ¶

Limit for magnitude of change in log10 eps_nuc at center. Only applies to increase in eps_nuc in non-convective core.. This can help to catch the start of core convection..

```
delta_log_eps_nuc_cntr_limit = 1
delta_log_eps_nuc_cntr_hard_limit = -1
```

`lg_XH_cntr` is log10(h1 mass fraction at center). Small timesteps as the center hydrogen is exhausted.

delta_lg_XH_cntr_min ¶

Ignore changes in `lg_XH_cntr` if value is less than this.

```
delta_lg_XH_cntr_min = -6
```

delta_lg_XH_cntr_max ¶

Ignore changes in `lg_XH_cntr` if value is more than this.

```
delta_lg_XH_cntr_max = -3
```

delta_lg_XH_cntr_limit ¶

If max delta is greater than this, reduce the next timestep by `delta_lg_XH_cntr_limit/max_delta`.

```
delta_lg_XH_cntr_limit = 0.05d0
```

delta_lg_XH_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XH_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XH_cntr_hard_limit = -1
```

`lg_XHe_cntr` is log10(he4 mass fraction at center) small timesteps as the center helium is exausted.

delta_lg_XHe_cntr_min ¶

Ignore changes in `lg_XHe_cntr` if value is less than this.

```
delta_lg_XHe_cntr_min = -6
```

delta_lg_XHe_cntr_max ¶

Ignore changes in `lg_XHe_cntr` if value is more than this.

```
delta_lg_XHe_cntr_max = -3
```

delta_lg_XHe_cntr_limit ¶

If max delta is greater than `delta_lg_XHe_cntr_limit`, reduce the next timestep by `delta_lg_XHe_cntr_limit/max_delta`.

```
delta_lg_XHe_cntr_limit = 0.1d0
```

delta_lg_XHe_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XHe_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XHe_cntr_hard_limit = -1
```

`lg_XC_cntr` is log10(c12 mass fraction at center). Small timesteps as the center carbon is exausted.

delta_lg_XC_cntr_min ¶

Ignore changes in `lg_XC_cntr` if value is less than this.

```
delta_lg_XC_cntr_min = -5
```

delta_lg_XC_cntr_max ¶

Ignore changes in `lg_XC_cntr` if value is more than this.

```
delta_lg_XC_cntr_max = -3
```

delta_lg_XC_cntr_limit ¶

If max delta is greater than `delta_lg_XC_cntr_limit`, reduce the next timestep by `delta_lg_XC_cntr_limit/max_delta`.

```
delta_lg_XC_cntr_limit = 0.1d0
```

delta_lg_XC_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XC_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XC_cntr_hard_limit = -1
```

`lg_XNe_cntr` is log10(ne20 mass fraction at center) Small timesteps as the center neon is exausted.

delta_lg_XNe_cntr_min ¶

Ignore changes in `lg_XNe_cntr` if value is less than this.

```
delta_lg_XNe_cntr_min = -5
```

delta_lg_XNe_cntr_max ¶

Ignore changes in `lg_XNe_cntr` if value is more than this.

```
delta_lg_XNe_cntr_max = 0
```

delta_lg_XNe_cntr_limit ¶

If max delta is greater than `delta_lg_XNe_cntr_limit`, reduce the next timestep by `delta_lg_XNe_cntr_limit/max_delta`.

```
delta_lg_XNe_cntr_limit = 1d99
```

delta_lg_XNe_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XNe_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XNe_cntr_hard_limit = -1
```

`lg_XO_cntr` is log10(o16 mass fraction at center) Small timesteps as the center oxygen is exausted.

delta_lg_XO_cntr_min ¶

Ignore changes in `lg_XO_cntr` if value is less than this.

```
delta_lg_XO_cntr_min = -5
```

delta_lg_XO_cntr_max ¶

Ignore changes in `lg_XO_cntr` if value is more than this.

```
delta_lg_XO_cntr_max = 0
```

delta_lg_XO_cntr_limit ¶

If max delta is greater than `delta_lg_XO_cntr_limit`, reduce the next timestep by `delta_lg_XO_cntr_limit/max_delta`.

```
delta_lg_XO_cntr_limit = 1d99
```

delta_lg_XO_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XO_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XO_cntr_hard_limit = -1
```

`lg_XSi_cntr` is log10(si28 mass fraction at center) Small timesteps as the center silicon is exausted.

delta_lg_XSi_cntr_min ¶

Ignore changes in `lg_XSi_cntr` if value is less than this.

```
delta_lg_XSi_cntr_min = -5
```

delta_lg_XSi_cntr_max ¶

Ignore changes in `lg_XSi_cntr` if value is more than this.

```
delta_lg_XSi_cntr_max = 0
```

delta_lg_XSi_cntr_limit ¶

If max delta is greater than `delta_lg_XSi_cntr_limit`, reduce the next timestep by `delta_lg_XSi_cntr_limit/max_delta`.

```
delta_lg_XSi_cntr_limit = 1d99
```

delta_lg_XSi_cntr_hard_limit ¶

If max delta is greater than `delta_lg_XSi_cntr_hard_limit`, retry with smaller timestep.

```
delta_lg_XSi_cntr_hard_limit = -1
```

limits based on changes in mass of the star ¶

delta_lg_star_mass_limit ¶

delta_lg_star_mass_hard_limit ¶

Limit for magnitude of change in log10(M/Msun).

```
delta_lg_star_mass_limit = 5d-3
delta_lg_star_mass_hard_limit = -1
```

limit for change in mdot in Msun/yr

- `delta_mdot_atol` tolerance for absolute changes
- `delta_mdot_rtol` tolerance for relative changes

```
delta_mdot_atol = 1d-3
delta_mdot_rtol = 0.5d0
```

delta_mdot_limit ¶

delta_mdot_hard_limit ¶

```
delta_mot = abs(mdot - mdot_old)/ (delta_mdot_atol*Msun/secyer + &
   delta_mdot_rtol*max(abs(mdot),abs(mdot_old)))
```

ignore if < 0

```
delta_mdot_limit = -1
delta_mdot_hard_limit = -1
```

factor_for_test_CpT_absMdot_div_L ¶

Limit on ratio `Cp(k)*T(k)*abs(mstar_dot)/L(k)` at `k = k_for_CpT_absMdot_div_L`. Cell index `k_for_CpT_absMdot_div_L` is set by the `adjust_mass` routine as follows: Let `delta_m` be `mdot*dt`, the change in mass for this step. Let `delta_m_for_limit = abs(delta_m)*factor_for_test_CpT_absMdot_div_L`. Then `k_for_CpT_absMdot_div_L` is the outermost cell boundary k, where the mass exterior to k is >= `delta_m_for_limit`.

```
factor_for_test_CpT_absMdot_div_L = 1
```

CpT_absMdot_div_L_limit ¶

Only use if > 0. Reduce next timestep if ratio is greater than this limit.

```
CpT_absMdot_div_L_limit = -1
```

CpT_absMdot_div_L_hard_limit ¶

Only use if > 0. Retry if ratio exceeds this limit.

```
CpT_absMdot_div_L_hard_limit = -1
```

limits based on changes in log total angular momentum ¶

delta_lg_total_J_limit ¶

If max delta is greater than `delta_lg_total_J_limit`, reduce the next timestep by `delta_lg_total_J_limit/max_delta`.

```
delta_lg_total_J_limit = 0.1d0
```

delta_lg_total_J_hard_limit ¶

If max delta is greater than `delta_lg_total_J_hard_limit`, retry with smaller timestep.

```
delta_lg_total_J_hard_limit = 0.5d0
```

limit_for_rel_error_in_energy_conservation ¶

hard_limit_for_rel_error_in_energy_conservation ¶

```
rel_error_in_energy_conservation = abs(error_in_energy_conservation/to
```

```
limit_for_rel_error_in_energy_conservation = 1d99
hard_limit_for_rel_error_in_energy_conservation = 1d99
```

limit_for_rel_rate_in_energy_conservation ¶

hard_limit_for_rel_rate_in_energy_conservation ¶

```
rel_rate_in_energy_conservation = abs(error_in_energy_conservation/tot
```

```
limit_for_rel_rate_in_energy_conservation = 1d99
hard_limit_for_rel_rate_in_energy_conservation = 1d99
```

limit_for_avg_lgE_residual ¶

hard_limit_for_avg_lgE_residual ¶

```
avg_lgE_residual = abs(dot_product(s% dq(1:s% nz),s% lnE_residual(1:s%
```

```
limit_for_avg_lgE_residual = 1d99
hard_limit_for_avg_lgE_residual = 1d99
```

limit_for_max_abs_lgE_residual ¶

hard_limit_for_max_abs_lgE_residual ¶

```
max_abs_lgE_residual = maxval(abs(s% lnE_residual(1:s% nz)))/ln10
```

```
limit_for_max_abs_lgE_residual = 1d99
hard_limit_for_max_abs_lgE_residual = 1d99
```

limit_for_avg_v_residual ¶

hard_limit_for_avg_v_residual ¶

```
avg_v_residual = abs(dot_product(s% dq(1:s% nz),s% v_residual(1:s% nz)
```

```
limit_for_avg_v_residual = 1d99
hard_limit_for_avg_v_residual = 1d99
```

limit_for_max_abs_v_residual ¶

hard_limit_for_max_abs_v_residual ¶

```
max_abs_v_residual = maxval(abs(s% v_residual(1:s% nz)))
```

```
limit_for_max_abs_v_residual = 1d99
hard_limit_for_max_abs_v_residual = 1d99
```

limit_for_abs_rel_E_err ¶

hard_limit_for_abs_rel_E_err ¶

```
abs_rel_E_err = maxval(abs(s% rel_E_err(1:s% nz)))
```

```
limit_for_abs_rel_E_err = 1d99
hard_limit_for_abs_rel_E_err = 1d99
```

limit_for_max_E_residual ¶

hard_limit_for_max_E_residual ¶

```
max_E_residual = maxval(abs(s% E_residual(1:s% nz)))
```

```
limit_for_max_E_residual = 1d99
hard_limit_for_max_E_residual = 1d99
```

report_why_dt_limits ¶

If true, produce terminal output about choice of timestep.

```
report_why_dt_limits = .false.
```

report_all_dt_limits ¶

If true, produce terminal output about all influences for choice of timestep.

```
report_all_dt_limits = .false.
```

report_hydro_dt_info ¶

If true, produce terminal output about choice of timestep based on `varcontrol_target`.

```
report_hydro_dt_info = .false.
```

report_dX_nuc_drop_dt_limits ¶

If true, report timestep limits from drop in abundance from nuclear reactions.

```
report_dX_nuc_drop_dt_limits = .false.
```

debugging controls ¶

report_hydro_solver_progress ¶

Set true to see info about newton iterations.

```
report_hydro_solver_progress = .false.
```

report_ierr ¶

If true, produce terminal output when have some internal error.

```
report_ierr = .false.
```

stop_for_NaNs ¶

If true and report_ierr is also true, then stop for NaNs.

```
stop_for_NaNs = .false.
```

trace_newton_bcyclic_solve_input ¶

Input is “B” j k iter B(j,k).

```
trace_newton_bcyclic_solve_input = .false.
```

trace_newton_bcyclic_solve_output ¶

Output is “X” j k iter X(j,k).

```
trace_newton_bcyclic_solve_output = .false.
```

trace_newton_bcyclic_matrix_input ¶

Matrix before factor.

```
trace_newton_bcyclic_matrix_input = .false.
```

trace_newton_bcyclic_matrix_output ¶

Matrix after factor.

```
trace_newton_bcyclic_matrix_output = .false.
```

trace_newton_bcyclic_steplo ¶

1st model number to trace.

```
trace_newton_bcyclic_steplo = 1
```

trace_newton_bcyclic_stephi ¶

Last model number to trace.

```
trace_newton_bcyclic_stephi = -1
```

trace_newton_bcyclic_iterlo ¶

1st newton iter to trace.

```
trace_newton_bcyclic_iterlo = 1
```

trace_newton_bcyclic_iterhi ¶

Last newton iter to trace.

```
trace_newton_bcyclic_iterhi = -1
```

trace_newton_bcyclic_nzlo ¶

1st cell to trace.

```
trace_newton_bcyclic_nzlo = 1
```

trace_newton_bcyclic_nzhi ¶

Last cell to trace; if < 0, then use nz as nzhi.

```
trace_newton_bcyclic_nzhi = -1
```

trace_newton_bcyclic_jlo ¶

1st var to trace.

```
trace_newton_bcyclic_jlo = 1
```

trace_newton_bcyclic_jhi ¶

Last var to trace; if < 0, then use nvar as jhi.

```
trace_newton_bcyclic_jhi = -1
```

To get info about the mesh set `show_mesh_changes = .true.`. Restart and get the `mesh_call_number` from terminal output. Set `mesh_dump_call_number = mesh_call_number`. Restart and it will write data files to `mesh_plot_data`. view with `test/mesh.rb` and `test/mesh_plan.rb`.

show_mesh_changes ¶

When `show_mesh_changes` is true, the terminal output includes the `mesh_call_number`.

```
show_mesh_changes = .false.
```

mesh_dump_call_number ¶

When `mesh_call_number == mesh_dump_call_number`, various plotting information is written..

```
mesh_dump_call_number = -1
```

trace_evolve ¶

Send evolve output to screen.

```
trace_evolve = .false.
```

variety of output from the hydro solver

hydro solver

```
hydro_numerical_jacobian = .false.
hydro_jacobian_nzlo = 1
hydro_jacobian_nzhi = -1
hydro_check_everything = .false.
hydro_inspectB_flag = .false.
hydro_sizequ_flag = .false.
hydro_get_a_numerical_partial = -1d0
hydro_test_partials_k = -1
hydro_show_correction_info = .false.
hydro_save_numjac_plot_data = .false.
hydro_dump_call_number = -1
hydro_dump_iter_number = -1
hydro_epsder_struct = 1d-5
hydro_epsder_chem = 1d-5
```

hydro_save_photo ¶

Saves a photo when hydro_call_number = hydro_dump_call_number - 1

```
hydro_save_photo = .false.
```

xa_clip_limit ¶

Abundances smaller than this limit are set to 0.

```
xa_clip_limit = 1d-99
```

trace_k ¶

Print out trace information about cell with number = `trace_k`.

```
trace_k = -1
```

fill_arrays_with_NaNs ¶

initialize arrays with NaNs to trap reads of uninitialized entries.

```
fill_arrays_with_NaNs = .true.
```

zero_when_allocate ¶

initialize arrays with zeros.

```
zero_when_allocate = .false.
```

miscellaneous controls ¶

relax_dY ¶

Change Y by this amount per step when relaxing Y.

```
relax_dY = 0.005d0
```

relax_dlnZ ¶

Change lnZ by this amount per step when relaxing Z. Default is ln10/10.

```
relax_dlnZ = 2.3025850929940459d-1
```

zams_filename ¶

Default is for Z=0.02, Y=0.28.

```
zams_filename = 'zams_z2m2_y28.data'
```

```
set_rho_to_dm_div_dV = .true.
```

use_other_{hook} ¶

Logicals to deploy the use_other routines.

```
use_other_eos = .false.
use_other_kap = .false.
use_other_atm = .false.
use_other_diffusion = .false.
use_other_mlt = .false.
```

```
use_other_adjust_net = .false.
use_other_adjust_mdot = .false.
use_other_j_for_adjust_J_lost = .false.
use_other_alpha_mlt = .false.
use_other_am_mixing = .false.
use_other_brunt = .false.
use_other_brunt_smoothing = .false.
use_other_build_initial_model = .false.
use_other_cgrav = .false.
```

```
use_other_energy_implicit = .false.
use_other_energy = .false.
use_other_momentum = .false.
use_other_eps_grav = .false.
use_other_mesh_functions = .false.
use_other_D_mix = .false.
```

```
use_other_neu = .false.
use_other_net_get = .false.
use_other_newton_monitor = .false.
use_other_opacity_factor = .false.
use_other_paquette_coefficients = .false.
use_other_pgstar_plots = .false.
use_other_porosity_factor = .false.
use_other_eval_fp_ft = .false.
use_other_torque = .false.
```

```
use_other_torque_implicit = .false.
use_other_eta_visc = .false.
use_other_wind = .false.
use_other_split_mix = .false.
use_other_after_struct_burn_mix = .false.
use_other_before_struct_burn_mix = .false.
use_other_surface_PT = .false.
```

```
use_other_export_pulse_data = .false.
use_other_get_pulse_data = .false.
use_other_edit_pulse_data = .false.
```

```
use_other_astero_freq_corr = .false.
```

mixing diffusion coeffs ¶

sig_term_limit ¶

Limit on coefficients in convective mixing equations. Consider a diffusion eqn of form:

```
x(k) - x0(k) = c1*(x(k-1) - x(k)) - c2*(x(k) - x(k+1))
```

Simplify for c1=c2=c, x(k-1)=x(k+1)=x0(k)=x0, x(k)=x0+dx Then eqn becomes

```
(1+2*c)*(x0+dx) - 2*c*x0 = x0
```

If `2*c >> 1`, then eqn becomes ill-conditioned, so we enforce `c <= sig_term_limit` In physical terms c is `dt*sig/dm`, where `sig = (4 pi r^2 rho)^2*D` and D = diffusion coeff (cm^2/s), so c can get large when dt/dm is large.

```
sig_term_limit = 1d13
```

am_sig_term_limit ¶

Limit on coefficients in angular momentum transport equations. Necessary for numerical stability. Plays same role as `sig_term_limit` for material mixing.

```
am_sig_term_limit = 1d13
```

sig_min_factor_for_high_Tcenter ¶

High center T limit to avoid negative mass fractions. If Tcenter >= `Tcenter_min_for_sig_min_factor_full_on`, then okay to reduce sig by as much as this factor as needed to prevent causing negative abundances. Inactive when >= 1d0.

```
sig_min_factor_for_high_Tcenter = 0.01d0
```

Tcenter_min_for_sig_min_factor_full_on ¶

If Tcenter >= this, factor = `sig_min_factor_for_neg_abundances`, this should be > `Tcenter_max_for_sig_min_factor_full_off`.

```
Tcenter_min_for_sig_min_factor_full_on = 3.2d9
```

Tcenter_max_for_sig_min_factor_full_off ¶

If Tcenter <= this, factor = 1, so has no effect this should be < `Tcenter_min_for_sig_min_factor_full_on`. For T > `full_off` and < `full_on`, factor changes linearly with Tcenter.

```
Tcenter_max_for_sig_min_factor_full_off = 2.8d9
```

max_delta_m_to_bdy_for_sig_min_factor ¶

sig_min factor goes to 1 as distance (in Msun units) from boundary of mixing region reaches this value

```
max_delta_m_to_bdy_for_sig_min_factor = 0.5d0
```

delta_m_upper_for_sig_min_factor ¶

okay to change sig min factor to 1 for mix region larger than this

```
delta_m_upper_for_sig_min_factor = 0.3d0
```

delta_m_lower_for_sig_min_factor ¶

don’t change sig min factor for mix region smaller than this

```
delta_m_lower_for_sig_min_factor = 0.1d0
```

Tcenter_max_for_dble_bcyclic ¶

if Tcenter <= this, use dble precision version of bcyclic. if Tcenter > this, use quad precision.

```
Tcenter_max_for_dble_bcyclic = 1d99
```

extra params as a convenience for developing new features note: the parameter `num_x_ctrls` is defined in `star_def.inc`

```
x_ctrl(1:num_x_ctrls) = 0d0
x_integer_ctrl(1:num_x_ctrls) = 0
x_logical_ctrl(1:num_x_ctrls) = .false.
```

One can split controls inlist into pieces using the following parameters. BTW: it works recursively, so the extras can read extras too.

read_extra_controls_inlist1 ¶

extra_controls_inlist1_name ¶

If `read_extra_controls_inlist1` is true, then read &controls from this namelist file.

```
read_extra_controls_inlist1 = .false.
extra_controls_inlist1_name = 'undefined'
```

If you try one of the following prebuilt extras, you must also set `read_extra_star_job_inlist1` true and change the `extra_star_job_inlist1_name` to match `extra_controls_inlist1_name`.

evolve 1 Msun from pre-ms to white dwarf

```
read_extra_controls_inlist1 = .true.
extra_controls_inlist1_name = 'inlist_extras_1M_lifecycle'
```

for debugging

```
extra_controls_inlist1_name = 'inlist_debug'
```

read_extra_controls_inlist2 ¶

extra_controls_inlist2_name ¶

If `read_extra_controls_inlist2` is true, then read &controls from this namelist file.

```
read_extra_controls_inlist2 = .false.
extra_controls_inlist2_name = 'undefined'
```

read_extra_controls_inlist3 ¶

extra_controls_inlist3_name ¶

If `read_extra_controls_inlist3` is true, then read &controls from this namelist file.

```
read_extra_controls_inlist3 = .false.
extra_controls_inlist3_name = 'undefined'
```

read_extra_controls_inlist4 ¶

extra_controls_inlist4_name ¶

If `read_extra_controls_inlist4` is true, then read &controls from this namelist file.

```
read_extra_controls_inlist4 = .false.
extra_controls_inlist4_name = 'undefined'
```

read_extra_controls_inlist5 ¶

extra_controls_inlist5_name ¶

If `read_extra_controls_inlist5` is true, then read &controls from this namelist file.

```
read_extra_controls_inlist5 = .false.
extra_controls_inlist5_name = 'undefined'
```