OP files and codes on CD

1 Introduction

The CD contains data files and codes required to obtain:

- OP monochromatic opacities for the chemical elements
  H, C, N, O, Ne, Na, Mg, Al, Si, S, Ar, Ca, Cr, Mn, Fe and Ni;
- Rosseland-mean opacities for any required chemical mixture;
- data for the calculation of radiative accelerations.

All files on the CD for monochromatic opacities are formatted and compressed (`gzip`).
The OP codes use binary data files (unformatted and not compressed).
Some initial processing is therefore required.

2 Contents of the CD

The CD contains a tar file `OPCD.x.y` where `x` and `y` specify version numbers. Running

```
tar xf OPCD.x.y
```

creates a directory `OPCD` containing this file `OPCD.ps`, a bash script `BASHOP`, a `makefile` and sub-directories:

- `codes`, all FORTRAN codes required;
- `mono`, all monochromatic opacities;
- `testin`, input files for testing the codes
- `testout`, outputs from test runs.

3 Installation of data from the CD

Installation can be done in one of three ways.

3.1 Use of `makefile`

Edit the file `makefile` and, on line 8, replace "`ifort`" with the name of the FORTRAN compiler to be used.
Enter

```
make
```

and then

```
make data
```

and then

```
make test.
```

3.2 Use of `BASHOP`

Enter

```
BASHOP
```

When prompted, enter the name of the FORTRAN compiler to be used.
When prompted, enter Y to continue tests or N to skip tests.

3.3 Manual installation

- extract all files from the `tar` file and `gunzip` all `.gz` files;
- compile the codes `unform.f`, `unforma.f` and `readop.f` and put the `.out` files produced in the directory `mono`;
- in `mono`, run `unform.out` which creates unformatted files `mzz.ttt` and `mzz.mesh` (for notation see section 4);
- in `mono` run `unforma.out` which creates unformatted files `azz.ttt` (see section 6);
- compile all other codes in the sub-directory `codes` and put the `.out` files in the directory `OPCD`;
- make test runs of the codes using the input files in `testin`;
- compare the output files created with those contained in the sub-directory `testout`

3.4 Timings and storage requirements

Use of `makefile` or `BASHOP` takes about 30 minutes on a 1.9 Hgz PC.
The size of the original tar file is 666 Mb.
The total data size after installation is 1604 Mb.
After installation the data files `fmzz.ttt.gz` and `fazz.ttt` can be deleted. The total size is then reduced to 941 Mb

4 Preliminaries

4.1 Ions

`iz`=nuclear charge Z.
`amamu`=atomic mass in atomic mass units.

4.2 Indices for temperatures and electron densities

$log(T) = 0.025*$`ite`
Range for `ite:  ite=ite1, ite2, ite3`
The minimum value for `ite` is `ite1`=140 ($log(T) = 3.5$) and the maximum value is `ite2`=320 ($log(T) = 8.0$).
$\log(N_e) = 0.25*$`jne`
Range for `jne`: see section 3.3 below.

4.3 Names of mono files

Unformatted files for mono-chromatic opacities have names `qzz.ttt` where
`zz` is a two-digit number for `iz` (`zz`=01 for H, `zz`=26 for Fe)
`ttt` is a three-digit number for `ite`.
`q` can be `f`, `m` or `c`: `f` is a fine mesh; `m` a medium mesh; and `c` (rarely used) a coarse mesh. The intervals are:

`q`	`ite3`	`jne3`	$\Delta\log(T)$	$\Delta\log(N_e)$
`f`	1	1	0.025	0.25
`m`	2	2	0.050	0.50
`c`	4	4	0.100	1.00

The range for $\log(N_e)$, (`jne1,jne2,jne3`), is specified on each file `qzz.ttt`.
The present CD contains only files for the mesh `q`=`m`.

4.4 Mono-chromatic and mean opacities

Notations used in the work of the Opacity Project are given in [2].
$u = h\nu/(kT)$.
For element k (specified by `iz`) the mono-chromatic opacity cross-section is $\sigma_k(u)$ in atomic units, a_0^2.
The stimulated emission factor, $[1 - \exp(-u)]$, is not included.
The total cross-section for a mixture is $\sigma(u) = \sum_k \sigma_k(u) f_k$ where f_k is the fractional abundance of k.

4.4.1 The Rosseland mean

The Rosseland mean cross-section is σ_{R},

$$\frac{1}{\sigma_{\mathrm{R}}} = \int_0^\infty \frac{F(u)}{\sigma(u)[1-\exp(-u)]}\,\mathrm{d}u$$

where

$$F(u) = [15/(4\pi^4)]u^4\exp(-u)/[1-\exp(-u)]^2.$$

All earlier OP calculations were made with `ntot` equally spaced points for u in a range `umin`$\leq u \leq$`umax`, and all earlier published opacity data were obtained with `umin`$=10^{-3}$, `umax`=20 and `ntot`$=10^4$.

4.4.2 The Planck mean

The Planck mean cross-section (rarely used) is

$$\sigma_{\mathrm{P}} = \frac{15}{\pi^4}\int_0^\infty \tilde{\sigma}(u)\frac{u^3}{[\exp(u)-1]}\,\mathrm{d}u$$

where $\tilde{\sigma}(u)$ does *not* include contributions from scattering processes.

4.5 Use of the $v-$mesh

$F(u)$ is small for u small ($F(u) \propto u^2$) and decreases exponentially for u large. New calculations have been made using equally-spaced points in the variable

$$v(u) = \int_0^u \frac{F(u)}{[1 - \exp(-u)]} \, \mathrm{d}u,$$

giving

$$\frac{1}{\sigma_\mathrm{R}} = \int \frac{1}{\sigma} \, \mathrm{d}v.$$

For a given `ntot`, use of v gives more points in regions where $F(u)$ is large and hence an improved frequency resolution (by a factor of about 3 in regions giving dominant contributions to σ_R).

All data on the CD are obtained using the v-mesh.

4.6 Units for Rosseland means

The codes compute Rosseland-mean cross-sections σ_R in atomic units (a_0^2).
Astronomers usually use Rosseland-mean opacities per unit mass,

$$\kappa_\mathrm{R} = \sigma_\mathrm{R}/\mu,$$

where μ is the mean atomic weight. As final output the codes give κ_R in cgs units, cm^2 g^{-1}.

4.7 Ionisation equilibria

`ne`=number of electrons in "target", as in the R-matrix work: the total number of electrons is (`ne`+1); `ne`= -1 for fully ionised; `ne=iz`-1 for neutral. The fraction in stage `ne` is `fion(ne)`.
All stages are included for `ne` in a range `ne1`$\leq$`ne`$\leq$`ne2` within which `fion(ne)`$\geq$`testi`.
`epatom`=number of electrons per atom: `epatom=iz` for fully ionised; `epatom`=0 for fully neutral.

4.8 Packing

Let the equally-spaced points in v be $v(n) = dv \times n$, and let `s(n)` be the value of $\sigma(u)$ at mesh-point n.
The packed data are stored in arrays `y(m)` and `nx(m)` with `m=1` to `np`. Point `m` corresponds to `n=nx(m)`, and `y(m)` is the corresponding value of `s(n)`. If [`nx(m+1)-nx(m)`]> 1, values of `s(n)` for intermediate value of `n` can be obtained by linear interpolations with fractional errors no larger than `opack`.
Data can be un-packed with the following code:-

```
b=y(1)
mb=1
s(1)=b
do m=2,np
  a=b
  b=y(m)
  ma=mb
```

```
    mb=nx(m)
    c=(b-a)/real(mb-ma)
    do k=1,mb-ma-1
      s(ma+k)=a+k*c
    enddo
    s(mb)=b
  enddo
```

Packed data require two numbers for each `m`, `nx(m)` and `y(m)`. If `np`>`ntot/2` packed data would take more storage than unpacked data: in that case packing is not used and `np` is set to 0.

4.9 Test runs

Input files for test runs are given in the sub-directory `testin`. Running the codes in `OPCD`, as in `BASHOP`, gives output files in `OPCD`. The tests are satisfactory if those files agree with the outputs provided in the sub-directory `testout`.

5 Opacity files and codes

5.1 Files `mzz.index`

Small formatted files `mzz.index` give information about all files `mzz.ttt` held in the same directory.
Their contents are:

`iz, amamu`
`ite1, ite2.  ite3`
`umin, umax`
`ncrse, ntot` (some data are calculated on a coarse mesh, `ncrse` points, then interpolated to the `ntot` mesh)
`opack`
`test1, testp, testl` used for testing inclusion of ionisation stages, partition function and spectrum lines
`optw4, optvdw, optwng` usually taken to be 1, 1, 1 for:-

- inclusion of the (W/WO)**4 factor in red wing profiles (see [2], section A4)
- inclusion of van der Waals broadening (important only when near-neutral)
- inclusion of line wings even when a line centre is at `u`>`umax`.

There are some differences in the files `mzz.index` for `zz`=01 and 02.
The files `mzz.ttt` are provided for `ite1`= 140, `ite2`= 320, `ite3`= 2 ($3.0 \leq \log(T) \leq 8.0$)

5.2 Files `mzz.mesh`

The mesh points in v are $v(n)$ =n*dv, n=1 to `ntot`.
The corresponding values of u are $u(n)$ =`umesh(n)`.
`umin=umesh(1)`, `umax=umesh(ntot)`.
`mzz.mesh` is a binary file. When placed on unit 5 it can be read with the code:-

```
parameter nptot=10000 (can be changed if need be)
dimension umesh(nptot)
read(5)dv,ntot,(umesh(n),n=1,ntot)
```

5.3 The opacity files `mzz.ttt`

An unformatted file `mzz.ttt` on unit 5 can be read with the following code:-

```
parameter(nptot=10000) (can be changed if need be)
dimension nx(nptot),y(nptot),s(nptot),fion(-1:27)
read(5)iz,ite,amamu,umin,umax,ncrse,nfine,opack,jne1,jne2,jne3
do j=jne1,jne2,jne3
  read(5)jne,epatom,oplnck,oross,ne1,ne2,(fion(ne),ne=ne1,ne2)
  read(5)np
  if(np.eq.0)then
     read(5)(s(n),n=1,ntot)
  else
     read(5)(nx(m),y(m),m=1,np)
  enddo
enddo
```

For unpacking, see section 3.8.

5.4 The code `readop.f`

Given `mzz` as input, the code `readop.f` reads all `mzz.ttt` files in a given directory. It produces two files with summary information, `mzz.smry` and `mzz.ion`.

`mzz.smry` gives values of `epatom, oplnck` and `oross` where `oplnck` and `oross` are Planck- and Rosseland-mean cross-sections. Placed on unit 5 it can be read with

```
read(5,*)iz,amamu,umin,umax,ncrse,ntot,opack
read(5,*)ite1,ite2,ite3
do i=ite1,ite2,ite3
  read(5,*)ite,jne1,jne2,jne3
  do j=jne1,jne2,jne3
    read(5,*)jne,epatom,oplnck,oross
  enddo
enddo
```

`mzz.ion` gives values of `fion`. Placed on unit 5 it can be read with

```
dimension fion(-1:27)
read(5,*)ite1,ite2,ite3
do i=ite1,ite2,ite3
  read(5,*)ite,jne1,jne2,jne3
  do j=jne1,jne2,jne3
```

```
      read(5,*)jne,ne1,ne2,(ne,fion(ne),ne=ne1,ne2)
    enddo
  enddo
```

5.5 The code `monop.f`

The code `monop.f` extracts a formatted opacity file for selected `ite` and `jne`. It prompts for: the input file name `mzz.ttt`; the required value of `jne`; an output file name. The output file on unit 5 can be read with

```
    dimension fion(-1:27)
    read(5,*)iz,ite,amamu,umin,umax,ncrse,ntot,opack
    read(5,*)epatom,oplnck,oross
    read(5,*)ne1,ne2
    read(5,*)(ne,fion(ne),ne=ne1,ne2)
 1  read(5,*,end=2)u,s
    goto 1
 2  stop
```

where `s` is the mono-chromatic opacity for frequency point `u`.

6 Mean opacities for mixtures

NOTE. There are some differences between the present codes for mixtures and versions circulated earlier. Users are advised to read the present documentation.

6.1 The code `mixv.f`

The code `mixv.f` differs from an earler code `mix3.f` in that it uses data on the $v-$mesh.
It uses files `mzz.index, mzz.mesh` and `mzz.ttt`.

Operation
For a specified mixture, the code `mixv.f` gives values of mass-density and Planck and Rosseland means on the entire array of (`ite,jne`) mesh points.
It obtains pressure-corrections to electron scattering, following [6]

Input
The following data are read from unit 5:-

`dir`, the directory for the input mono files
`outfle`, name of the output file
`X`, hydrogen mass-fraction
`Z`, metals mass fraction
`nel`, number of elements included
list of values of `iz` and `fa` for metals
`ite1, ite2, ite3` for range of temperatures to be included.

Note:-

- The directory name `dir` should end with "/". It may be of any length consistent with `character*100`. Leading or trailing blanks are stripped by the code.
- For each value of `iz`, `fa` gives the relative abundance of a metal. Normalisation of the `fa` is immaterial: re-normalisation is done internally.

Output
For each `ite, jne` the output file give values of

`flrho`= $log(\rho)$, $\rho =$ mass density in g cm^{-3}
`planck`= Planck mean in cm^2 g^{-1}
`ross`= Rosseland mean in cm^2 g^{-1}

The output file, placed on unit 5, can be read with the code

```
      character head*70
      dimension iz(17),fa(17)
   70 format(a70)
      read(5,70)head
      read(5,*)nel,ite1,ite2,ite3
      do n=1,nel
        read(5,*)iz(n),fa(n)
      enddo
      do i=ite1,ite2,ite3
        read(5,*)ite,jne1,jne2,jne3
        do j=jne1,jne2,jne3
          read(5,*)jne,flrho,planck,ross
        enddo
      enddo
```

The output from `mixv.f` has the same form as that from the older code `mix3.f`.

Test run
Input: `in.mixv` (s92 mixture (see [2]));
Output: `mixv.s92`

6.2 The code `opfit.f`

Operation
`opfit.f` gives interpolated values of $\log(\kappa)$ where κ is either $\kappa_{\rm P}$ (Planck mean) or $\kappa_{\rm R}$ (Rosseland mean), in units of cm^2 g^{-2}.
Its operation, involving the use of bi-cubic spline interpolations, is described in [1]
It has options for two types of output.

1. Tables of $\log(\kappa)$ in OPAL format (see [8]).
2. Values of $\log(\kappa)$ and its derivatives interpolated to input values of $\log(T)$ and $\log(\rho)$

Input
The current version of `opfit.f` differs from earlier versions only in the organisation of input from unit 5.
The input is: -

> name of the file `outfle` from run of `mixv.f`
> name of the output file
> `ism` for number of passes through a smoothing filter
> `iop` = 0 for Rosseland mean, 1 for Planck mean
> `iopal` = 0 for interpolations, 1 for tables in OPAL format
> For `iopal=0`, a sequence of values of `flt`= $\log(T)$ and `flrho`= $\log(\rho)$

It is assumed that the input and output files are in the current working directory.

Output
Output is either:-

1. For `iopal`= 1, a table of opacities in OPAL format.
2. For `iopal`= 0, a list of values of:-

    `flt, flrho, G, DGDT,DGDRHO`
    (`G`=$\log(\kappa_{\rm R})$, `DGDT`=$\partial\log(\kappa_{\rm R})/\partial\log(T)$; `DGDRH`=$\partial\log(\kappa_{\rm R})/\partial\log(\rho)$).

Test run
Input: `in.opfit` (s92 mixture, $\log(R) = -1.75$)
Output: `opfit.s92`

6.3 The code `mx.f`

Operation
The code `mx.f` obtains opacities in a one-step process using bi-cubic interpolations without splines.

For a function of one variable, 4 points are required for cubic interpolations; for a function of two variables, 16 points are required. For the first (T, ρ) pair mixture opacities are calculated on the 16 required values of (`ite, jne`) and interpolations are made. For the next pair a different set of 16 points may be required: new calculations are made only for points not used for the previous pair.
Corrections to scattering are made, as in `mixv.f`

Input
The input for `mx.f` is :-

> (1) `dir`, directory for mono files, as in `mixv.f`.
> (2) `q`, specifying mesh-type (`f, m` or `c`)
> (3) Names of two files, for output of Rosseland means and of abundances.
> (4) `nel`= number of elements,
> `(iz(n),fa(n))` for nuclear charges and abundances.
> (5) A sequence of values of values of `flt`= $\log(T)$, `flrho`= $\log(\rho)$ and `newa`.
> (6) A new set of abundances is signalled by `newa` not equal to zero. An entry (4) with `newa`$\neq 0$ is followed by a repeat of step (3)

Output
Output is:-

> (1) A file written to unit 7 giving values of `npoint`, `flt`, `flrho`, `newa`, `G`$= \log(\kappa_R)$, `DGDT` and `DFDRHO` (`npoint` is the sequence number for the (T, ρ) point).
> (2) A file written to unit 8 only for `newa`$\neq 0$, giving values of `npoint`, `nel`, `flt`, `flrho` and (`iz(n), fa(n),n=1,nel`).

The output files are formatted and are self-explanatory.

Test run

Input:	`in.mx` (same case as for `opfit.f`)
Output:	`mx.s92.g` (unit 7 data)
	`mx.s92.a` (unit 8 data)

6.4 The code `mixz.f`

If the same metal-mixture is to be used many times, for different stellar models or for different values of X and Z, it can be convenient to compute and store the monochromatic opacities for that metal-mixture. That can be done using the code `mixz.f`.

Operation
Operation of `mixz.f` is similar to that of `mixv.f`. The monochromatic opacity files are stored as a "fictitious" metal: `mzz.ttt` where `zz` is a two-digit number larger than 28.

Input

> (1) Names of directories: `diri` for the single-element mono opacities to be used; `diro` for the output mixture mono opacities.
> (2) `izm`, the two-digit number `mzz`
> (3) `nel`, the number of metals in the mixture.
> (4) `(iz(n), fa(n),n=1,nel)`, nuclear charges and abundance.
> (5) `ite1,ite2,ite3`, temperature range.

Output
Output files `mzz.index, mzz.smry, mzz.ttt` in the directory `diro`.

Test run
`in.mixz`, the case of the metal mixture `a04` defined in [5].
Produces mixture mono opacities `m30.ttt`.

6.5 The code `mxz.f`

The code `mxz.f` is similar to `mx.f` but uses mixture mono opacities for metals previously computed with `mixz.f`. Uses bi-cubic interpolations, not splines.

Input

> (1) `dir`, the directory to be used for mono opacities of H, He and the metal-mixture.
> (2) `mzz`, the label for the metal mixture (character*3).
> (3) Names of two output files: one for mean opacities, one for information on X and z.

(4) Values of `flt`= $\log(T)$, `flr`= $\log(\rho)$ and `newa`
(5) For `newa`$\neq 0$, values of X and Z.

Output

File on unit 7:

(1) `npoint`
(2) `flt, flrho, newa`
(3) `G`= $\log(kr)$, `DGDT`= $\partial\log(\kappa_{\rm R})/\partial\log(T)$, `DGDRH`= $\partial\log(\kappa_{\rm R})/\partial\log(\rho)$.

File on unit 8:

`npoint, flt, flrho`, X and Z.

Test Run
`in.mxz`, results for the solar model `bp04` (see [5]) with X and Z varying continuously in the radiative interior.

6.6 Choice of mixture codes

The Rosseland mean opacities, `G`= $\log(\kappa_{\rm R})$, are required for calculations of stellar structures and the derivatives, `DGDT`= $\partial\log(kr)/\partial\log(T)$ and `DGDRHO`= $\partial\log(\kappa_{\rm R})/\partial\log(\rho)$, are required for studies of stellar pulsations.

- `mixv` provides mixture opacities for all mesh-points on the entire (`ite, jne`) plane. It can be convenient if results are subsequently to be computed, using `opfit.f`, for a large number of (T,ρ) values and all for the same mixture. The use of spline interpolations ensures that the derivatives are always continuous. If required, some smoothing can be done using the parameter `ism` but with the present data that should rarely, if ever, be needed.

- `mx.f` is faster because it requires calculation of mixture opacities only for those (`ite,jne`) points required for cubic interpolations. Its use is convenient when results are required for many different mixtures. However, `mx.f` does not ensure continuity of the derivatives, and it provides no facilities for smoothing.

- `mxz.f` is convenient for cases with X and Z varying with depth.

The test runs of `opfit.f` and `mx.f` are for the same case, S92 mix, $\log(R) = -1.75$, $\log(T) = 6.0$ to 7.3. Comparison of the files produced, `opfit.s92` and `mx.s92.g`, shows close agreement for `G`= $\log(\kappa_{\rm R})$ and `DGDT`= $\partial\log(kr)/\partial\log(T)$ but some lack of smoothness in `DGDRHO`= $\partial\log(kr)/\partial\log(\rho)$ from `mx.f`: see Figure 1 of [7].

7 Radiative accelerations

NOTE. There are some differences between the present codes for radiative accelerations and versions circulated earlier. Users are advised to read the present documentation.

The radiative acceleration in a star for element k is (see [3])

$$g_{\rm rad}(k) = (1/c)[\mu/\mu(k)]\mathcal{F}\kappa_{\rm R}\gamma(k)$$

where:

c is the speed of light;
μ is the mean atomic weight for a mixture, $\mu(k)$ the atomic weight for element k;
$\mathcal{F}$ is the total radiative flux:

$\mathcal{F} = \pi B(T_{\text{eff}})(R_\star/r)^2$;
T_{eff} is the effective temperature;
$B(T) = 2(\pi k_{\text{B}}T)^4/(15c^2h^3)$;
k_{B} is Boltzmann's constant;
r is the distance from the centre of the star and $R_\star$ the radius of the star.

$\gamma(k)$ is the dimensionless quantity

$$\gamma(k) = \int \frac{\sigma_k^{\text{mta}}}{\sigma}\,\mathrm{d}v.$$

$\sigma_k^{\text{mta}}(u)$ is the cross-section for momentum transfer to an atom (see [3]),

$$\sigma_k^{\text{mta}}(u) = \sigma_k(u) \times [1 - \exp(-u)] - a_k(u)$$

where subtraction of $a_k(u)$ takes out contributions from electron scattering and from momentum transfer to electrons.

It is necessary to specify the abundances of all elements contributing to σ. Use is made of an abundance-multiplier, χ, such that the abundance of the selected element, k, can be multiplied by χ, leaving the relative abundances of all other elements unchanged.

Derivatives with respect to χ, $\partial\kappa_{\text{R}}/\partial\chi$ and $\partial\gamma/\partial\chi$, can be useful in making interpolations in χ.

A quantity ζ, defined in section 2.10 of [3], is used for the calculation of diffusion coefficients.

7.1 The files `azz.ttt`

The cross-sections a_k are given in the files `azz.ttt`, which have structures similar to those of the mono-opacity files `mzz.ttt`. They can be read using the code `mona.f`.

7.2 The code `accv.f`

The code `accv.f` has a structure similar to that of `mixv.f` (see section 4.1). Data are read from the files `mzz.ttt` and `azz.ttt` and values of

`ross`=κ_{R}, `rossp`=$\partial\kappa_{\text{R}}/\partial\chi$
`gam`=γ, `gamp`=$\partial\gamma/\partial\chi$

are obtained on the (T, N_e) mesh points.

Input

(1) `dir`, directory for the files `mzz.ttt`, `azz.ttt`
(2) `nel`, the number of elements
(3) `(iz(n), fa(n), n=1,nel)`, nuclear charges and abundances
(4) `iz1`, nuclear charge for the selected element
(5) `nchi` (number of χ values), `chilo` (lowest value of $\log(\chi)$), `dchil` (the interval in $\log(\chi)$

Output
A file `acc.zz` where `zz` is a two-digit number for `iz1`, the selected element.

(1) `iz1, ntot, nchi, fmu0, fmu1`. The mean atomic weight is `fmu`=`fmu0`+`fmu1`$\times\chi$
(2) `amacc, fanacc`, mass of selected element and its abundance for $\chi = 1$.
(3) `chi(l),l=1,nchi`
(4) `ite1,ite2,ite3`, range for temperature index
Start loop on `ite`:

> (5)`ite,jn1,jn2,jn3`. `ite` and range for `jne`
> Start loop on `jne`
>
> > (6) `jne,epa0,epa1`. Number of electrons per atom is `epa`= `epa0`+`epa1`$\times\chi$
> > (8) `zet` used for diffusion coefficient
> > (9) `(oross(l),l=1,nchi)`. Values of $\kappa_{\rm R}$
> > (10)`(orossp(l),l=1,nchi)`. Values of $\partial\kappa_{\rm R}/\partial\chi$
> > (11)`(gam(l),l=1,nchi)`. Values of γ
> > (12)`(gamp(l),l=1,nchi)`. Values of $\partial\gamma/\partial\chi$

(13) `nel`
(14) (izz(n),fa(n),n=1,nel)

The file output from `accv.f` can be read with the code `readacc.f` which includes full documentation.
Test run
Input: `in.accv`. s92 mix, `iz1`=18 (selected element Argon).
Output: `acc.18`.

7.3 The code `accfit.f`

The code `accfit.f` has a structure similar to that of `opfit.f` (see section 4.2). It uses data from the file `acc.zz` output from `accv.f` (`subroutine read1` of `accfit.f` has comments on the contents of that file).
Note. The mesh for χ used in `accfit.f` need not be the same as that used in `accv.f` (it maybe convenient to use a finer mesh in `accfit.f`): interpolations to the χ-values for `accfit.f` are made using the derivatives, $\partial\kappa_{\rm R}/\partial\chi$ and $\partial\gamma/\partial\chi$ provided on the output file from `accv.f`.
input
List of names of 4 files:

(1) `file1`, the file `acc.zz` output from `accv.f`
(2) `file2`, value of $T_{\rm eff}$ for a stellar model and values of $\log(T)$, $\log(\rho)$ and $(r/R_\star)$ for depth points
(3) `file3`, values of `chilo` (lowest value of $\log(\chi)$), `dchil` (interval in $\log(\chi)$), `mchi` (number of values of $\log(\chi)$)
(4) `file4`, name of the output file.

Output

(1) `nstar,chilo,dchil,nchi`. `nstar` is number of depth points in a stellar model.
Start loop on depth points `n`:

(2) `n,flt,flrho`. Depth point and $\log(T)$, $\log(\rho)$
(3) `(zetout(l),l=1,nchi)`. ζ used for diffusion coefficient.
(4) `(rssout(l),l=1,nchi)`. $\log(\kappa_{\rm R})$
(5) `(gmmout(l),l=1,nchi)`. $\log(\gamma)$
(6) `(grdout(l),l=1,nchi)`. $\log(g_{\rm rad})$

Test run
Input: `in.accfit`. 92 mixture, `iz1`=18 (Argon), stellar model $T_{\rm eff} = 10^4$
Output: `accfit.18`

7.4 The code `ax.f`

Operation of the code `ax.f` is similar to that of `mx.f` (see section 4), using bi-cubic interpolations.
Input

(1) `dir`, name of the directory for the files `mzz.ttt` and `azz.ttt`;
(2) `nch` (the number of χ values), `chilo` (the smallest value of $\log(\chi)$, `dchil` (the interval in $\log(\chi)$;
(3) `q`, mesh type;
(4) names of two output files, to be attached to units 7 and 8;
(4) `nel`, total number of elements in mixture;
(5) `iz1`, nuclear charge for selected element;
(6) `(iz(n),fa(n),n=1,nel)`, nuclear charges and abundances;
(7) value of $T_{\rm eff}$ for a stellar model and values of `flt, flrho,` $r/R_\star$, and newa for each depth point (`flt`= $\log(T)$, `flrho`= $\log(\rho)$)
(8) `newa`$\neq 0$ read in step (7) signals new abundances and is followed by a repeat of step(5)

Output

`npoint` is the sequence number for depth points, starting `npoint`= 1.
`zet` is the quantity ζ which is defined in [3] and is used in the calculation of diffusion coefficients.

`zet`
`g`= $\log(\kappa_{\rm R})$
`gp`= $d\log(\kappa_{\rm R})/d\log(\chi)$
`f`= $\log(\gamma)$
`fp`= $d\log(\chi)/d\log(\chi)$
`grl`= $\log(g_{\rm rad})$

(a) Output to unit 7 for each depth point
`npoint, flt, flrho, newa`
Then for each depth point, as functions of $\chi(l)$, $l = 1$, `nch`

```
(zet(l),l=1,nch)
(g(l),l=1,nch)
(gp(l),l=1,nch)
(f(l), l=1,nch)
```

```
        (fp(l),l=1,nch)
        (grl(l),l=1,nch)
```

(b) Output to unit 8 for `newa`$\neq 0$

```
npoint, flt, flrho
(iz(n), fa(n), n=1,nel)
```

Test run

Input:	`in.ax`. As for `accfit`
Output:	`ax.18.g`
	`ax.18.a`

7.5 Interpolations of acceleration data

The quantities κ_{R} and γ are interpolated in T, ρ and χ. The interpolations are best made using $g = \log(\kappa_{\mathrm{R}})$ and $f = \log(\gamma)$, which are the quantities employed in `ax.f`. However, in a few rare circumstances one obtains $\gamma \leq 0$ and $\log(\gamma)$ is no longer a real number. In those circumstances, the code give output values of f and fp to be equal to -30.

The derivatives with respect to $\log(\chi)$ are intended to aid subsequent interpolations in χ. Given $y(x)$ and $\mathrm{d}y(x)/\mathrm{d}x$ at $x = x_1$ and x_2, $y(x)$ can be fitted to a cubic for $x_1 \leq x \leq x_2$.

7.6 Choice of acceleration codes

The one-step process of using `ax.f` is adequate for most purposes and is convenient for cases in which compositions can change as functions of depth in a star. The two-step process, `accv.f` and `accfit.f` can be used to obtain improved accuracy in difficult cases.

The test runs of `accfit.f` and `ax.f` are for the same case, Argon in a stellar model with $T_{\mathrm{eff}} = 10^4$. Comparisons of results for g_{rad} shows some lack of smoothness in values from `ax.f` for the extreme case of $\chi = 0.01$: see Figure 2 of [7].

8 Summary on files and codes

8.1 Data files

8.1.1 Compressed formatted data files from tar file

```
fmzz.ttt.gz     Monochromatic opacities for element zz, temperature ttt. 546.7 Mb
fmzz.mesh.gz    Information on frequency mesh. 0.44 Mb
fazz.ttt.gz     Monochromatic data required for radiative accelerations. 117.7 Mb
```

8.1.2 Unformatted data files after processing

```
mzz.ttt 708.2 Mb
mzz.mesh 0.68 Mb
azz.ttt  159.1 Mb
```

8.2 Codes

8.2.1 Codes for mean opacities (see section 6)

`mixv, opfit, mx, mxz, mixz`

8.2.2 Codes for accelerations (see section 7)

`accv, accfit, ax`

8.3 Codes for processing data from `tar` file

- Codes used by `BASHOP`, `unform _ bash, unforma _ bash, mesh _ bash`
- Codes not used by `BASH`, `unform, unforma, readop`

8.4 Other codes

`monop`, see section 5.5
`readacc`, see section 7.2
`form` and `forma` create formatted files from unformatted files.

References

References

[1] M.J. Seaton, "Fitting and smoothing of opacity data", MNRAS, **265**, L25, 1993

[2] M.J. Seaton, Yu Yan, D. Mihalas, A.K. Pradhan, "Opacities for stellar envelopes", MNRAS, **266**, 805, 1994

[3] M.J. Seaton, "Radiative accelerations in stellar envelopes", MNRAS, **289**, 700, 1997

[4] N.R. Badnell, M.J. Seaton, "On the importance of inner-shell transitions for opacity calculations", J. Phys. B, **36**, 4367, 2003

[5] N.R. Badnell, M.A. Bautista, K. Butler, F. Delahaye, C. Mendoza, P. Palmeri, C.J. Zeippen, M.J. Seaton, "Up-dated opacities from the Opacity Project", MNRAS, submitted

[6] D.B. Boerckner, ApJ, 316, L98, 1987

[7] M.J. Seaton, MNRAS, to be submitted, 2004

[8] F.J. Rogers, C.A. Iglesias, 1992, ApJS , 79, 507