Directions for the radiative transfer code MOLLIE

Basic system requirements

MOLLIE runs in parallel using MPI the message passing interface. We usually use the latest version, MPICH2 from Argonne National Laboratory. This is the most commonly used parallel implementation, freely available and support by ANL. We have used other MPI-style implementations such as LAM MPI and open MPI.

MOLLIE cannot be run without some version of MPI.

MOLLIE runs on Apple laptops as well as Linux clusters as long as MPI is installed.

If you have a cluster for parallel computing, MPI is surely already installed, probably several versions. You have to find out how to run MPI on your cluster. Usually there is a queuing system, the script will require some particular shell commands.

MPI is easy to install on Debian-based Linux systems such as Ubuntu using the package manager synaptic or the command line apt-get. Install the package MPICH2 and its dependencies. Make sure that gfortran is installed. This is the newer version of g77. (There is probably an RPM for the other Linux.)

MOLLIE requires at least two processes, one master and at least one slave. These can be on the same processor although almost all computers, even laptops, now have multiple processors.

The Lambda iteration (L-iteration) has two steps, solution of the statistical equilibrium equations for the level populations and solution of the radiative transfer for the mean radiation field. The solutions for statistical equilibrium are run on the master processor which distributes the information for the slaves to run the ray tracing. Unlike some parallel codes, the memory requirements are not distributed. The master process needs enough memory to hold the entire model cube and the level populations. The slaves process one ray at a time and require very little memory. The radiative transfer solution scales very well with the number of processors. The computation of the statistical equilibrium equations is not parallelized, but is always faster than the ray tracing. Still, large models with grid sizes > 100^3 will take some time. Small models of 32^3 generally run in a few seconds per L-iteration.

The memory required to hold the input model is about the number of cells, times the number of bytes per variable, times the number of variables. In total, the number of bytes required is about 100 times the number of cells. The memory required to hold the output is the square of the number of cells in one dimension times the number of channels in the spectra, times the number of spectral lines, times the number of bytes per variable.

Some parts of MOLLIE use F77 that does not allow for dynamic memory allocation. There are some hard-coded limits on the F77 array dimensions that can be changed if necessary.

MOLLIE has been run with models of 512^3 , but is slow. In general, it is better to use nested grids, several levels of 128 for example. (The grid size does not need to be a power of two unless the option for beam convolution is selected. The output can always be convolved with a beam in another program.)

MOLLIE contains both fortran and C programs. The compilers, defined at the time MPICH is installed, need to be a matched pair, for example, both GNU compilers or both Intel compilers. Intel is supposed to be compatible with GNU, but mixing has never been tried. If the compilers are not matched, the memory space of the executable will be confused resulting in run-time errors that are impossible to debug.

The output of MOLLIE is a binary file, ModelCube0, that is read by an IDL program, load_model.pro. The binary file could be read by a program in any other language. The code to do so could be translated from the IDL program, but does not yet exist. An easy way to use other programs to visualize the output is to read MOLLIE's output and use the IDL FITS writing program from GSFC's FITS library to write a FITS file after selecting a single transition and viewing angle in the output. This is a subroutine call of one line of programming.

Model Definition

The basic model is defined in terms of the density, temperature, 3D velocity, line width, and abundance of the molecular tracer. A more complex model can include the continuum emission from dust or ionized gas. MOLLIE also has internal modules to define the molecular abundance with some simple models of chemistry and photodissociation. These are applicable only to starless cores.

The model is defined in the C program **define_model.c** and stored in the structure model.

```
struct model {
        float temperature;
        float density;
        float vx,vy,vz;
        float abundance;
        float linewidth;
        int adds;
        int phase;
        float mean_av;
        float dust_temperature;
};
```

```
extern struct model ****model;
```

Most of the variable names are obvious. CGS units are used everywhere in MOLLIE except for linear distances which are in pc.

The variable “adds” counts the number of rays in each cell and is not an input variable. Ignore this when defining the model.
The variable “phase” refers to either molecular (0) or ionized gas (2). There is no implementation for neutral (1) gas. Defaults to molecular.
The “mean_av” is in fact the transmission, exp(-Av). This is calculated by the photo-dissociation and chemistry modules and is not an input. Ignore this when defining the model.

The essential purpose of the function **define_model** is to fill in the structure **model**. Other structures and functions in the file define_model.c are used in special cases. For example, the structure “dcube” holds the output, the molecular spectra. It is used only if the automatic model fitting is used. The functions whose names begin with “read_” were written to read the output of various hydro codes that we have used in the past. For the current L1544 project, read_lynds is used.

Look in the file define_model.c for the main loop to define the model:

```
for (ibox=0;ibox<nbox;ibox++) {
 for (i=0;i<nx[ibox];i++) {
 for (j=0;j<ny[ibox];j++) {
 for (k=0;k<nz[ibox];k++) {

    x = xcenter[ibox][i];
    y = ycenter[ibox][j];
    z = zcenter[ibox][k];
    idir = 1;
    crtsph_(&x,&y,&z,&r,&az,&el,&idir);
    sinaz = sin(az);
    sinel = sin(el);
    cosaz = cos(az);
    cosel = cos(el);
```

There are 4 loops, the outermost, ibox, is the level of nested grids. The other 3, i,j,k, are the Cartesian coordinates for each nested level. The number of boxes and the linear dimensions of the grid, in parsec, are defined in setup.c. The 9 lines below are for convenience in defining the model. For each cell, ibox,i,j,k, both the Cartesian coordinates x,y,z are defined for the center of each cell, the corresponding polar coordinates, r,az,el, are defined by the function crtsph, and the sines and cosines of the polar angles. With this information, the model cube can be filled in either from simple equations or by reading in a hydrodynamic model.

For example, here is the definition of temperature using a linear interpolation of the temperature from a hydro model. The temperature, `hst`, is defined on a radial grid, `hsr`, both of which were filled in earlier. The variables `jlo` and `jhi` define the indices of the hydro array on either side of the radial coordinate, `r`, of the cell center.

```
model[ibox][i][j][k].temperature = hst[jlo] +
(r - hsr[jlo])*(hst[jhi]-hst[jlo])/(hsr[jhi]-hsr[jlo]);
```

The temperature could also be defined as a power law if we do not have a hydro model, after defining T0 and r0 earlier in the program.

```
model[ibox][i][j][k].temperature = T0 + pow(r0/r,0.25)
```

temperature Here is another example showing how the line width is defined, using the previously defined temperature.

```
/* Here the linewidth is the microturbulent linewidth added
in quadrature to the thermal linewidth */

        model[ibox][i][j][k].linewidth   =
                sqrt(turbwidth*turbwidth
                  + BOLTZ* model[ibox][i][j][k].temperature
                     /(atoms*AMU));
```

When the code is run, the output file, **Cnotes0**, shows the model along the 3 axes through the center. This shows the model as MOLLIE sees it. It's always a good idea to check the definition of model as understood by MOLLIE.

Here is the output from Cnotes0 showing the model along the Z axis through the center, i,j = 16,16 for the first, outermost nested level. Because there are an even number of cells, the cell centers are ½ cell off the origin. The table below looks better in Cnotes0 without the line-wrapping.

```
Model defined over box 0 with nx,ny,nz = 32 32 32
Model defined over box 1 with nx,ny,nz = 32 32 32
Model defined over box 2 with nx,ny,nz = 32 32 32
Model defined
Box number 0 out of 3
Number of cells 32
Check the model at i,j = 16 16 x,y = 0.010000 0.010000
  k      z       T     Tdust  density       vx         vy         vz     linewidth  mean
Av    abundance
  0 -0.310000  18.45  14.76  1.386e+02  6.201e+00  6.202e+00 -1.922e+02  1.081e+04
7.588e-01  2.376e-12
  1 -0.290000  17.66  14.64  1.565e+02  1.811e+00  1.811e+00 -5.251e+01  1.071e+04
7.285e-01  2.989e-12
  2 -0.270000  16.84  14.57  1.789e+02 -4.082e+00 -4.082e+00  1.102e+02  1.060e+04
6.985e-01  3.811e-12
  3 -0.250000  15.97  14.38  2.079e+02 -1.280e+01 -1.280e+01  3.201e+02  1.048e+04
6.616e-01  5.085e-12
```

```
  4 -0.230000  15.03  14.20  2.467e+02 -2.608e+01 -2.608e+01  5.998e+02  1.035e+04
6.231e-01  6.991e-12
  5 -0.210000  14.03  14.02  3.010e+02 -4.669e+01 -4.669e+01  9.805e+02  1.021e+04
5.819e-01  1.002e-11
  6 -0.190000  12.86  13.69  3.832e+02 -7.804e+01 -7.804e+01  1.483e+03  1.004e+04
5.313e-01  1.559e-11
  7 -0.170000  11.32  13.46  5.289e+02 -1.264e+02 -1.264e+02  2.149e+03  9.821e+03
4.830e-01  2.571e-11
  8 -0.150000   9.27  13.04  8.378e+02 -2.029e+02 -2.029e+02  3.043e+03  9.518e+03
0.000e+00  2.000e-10
  9 -0.130000   8.23  12.50  1.363e+03 -3.023e+02 -3.023e+02  3.930e+03  9.358e+03
0.000e+00  2.000e-10
 10 -0.110000   8.16  11.83  2.223e+03 -4.222e+02 -4.222e+02  4.644e+03  9.348e+03
0.000e+00  2.000e-10
 11 -0.090000   8.65  11.08  3.686e+03 -6.034e+02 -6.034e+02  5.431e+03  9.423e+03
0.000e+00  2.000e-10
 12 -0.070000   9.32  10.18  6.532e+03 -9.116e+02 -9.116e+02  6.381e+03  9.525e+03
0.000e+00  2.000e-10
 13 -0.050000  10.00   9.24  1.305e+04 -1.506e+03 -1.506e+03  7.529e+03  9.627e+03
0.000e+00  2.000e-10
 14 -0.030000  10.47   8.14  3.200e+04 -2.991e+03 -2.991e+03  8.973e+03  9.697e+03
0.000e+00  2.000e-10
 15 -0.010000   9.89   7.22  1.114e+05 -7.671e+03 -7.671e+03  7.671e+03  9.610e+03
0.000e+00  2.000e-10
 16  0.010000   9.89   7.22  1.114e+05 -7.671e+03 -7.671e+03 -7.671e+03  9.610e+03
0.000e+00  2.000e-10
 17  0.030000  10.47   8.14  3.200e+04 -2.991e+03 -2.991e+03 -8.973e+03  9.697e+03
0.000e+00  2.000e-10
 18  0.050000  10.00   9.24  1.305e+04 -1.506e+03 -1.506e+03 -7.529e+03  9.627e+03
0.000e+00  2.000e-10
 19  0.070000   9.32  10.18  6.532e+03 -9.116e+02 -9.116e+02 -6.381e+03  9.525e+03
0.000e+00  2.000e-10
 20  0.090000   8.65  11.08  3.686e+03 -6.034e+02 -6.034e+02 -5.431e+03  9.423e+03
0.000e+00  2.000e-10
 21  0.110000   8.16  11.83  2.223e+03 -4.222e+02 -4.222e+02 -4.644e+03  9.348e+03
0.000e+00  2.000e-10
 22  0.130000   8.23  12.50  1.363e+03 -3.023e+02 -3.023e+02 -3.930e+03  9.358e+03
0.000e+00  2.000e-10
 23  0.150000   9.27  13.04  8.378e+02 -2.029e+02 -2.029e+02 -3.043e+03  9.518e+03
0.000e+00  2.000e-10
 24  0.170000  11.32  13.46  5.289e+02 -1.264e+02 -1.264e+02 -2.149e+03  9.821e+03
4.830e-01  2.571e-11
 25  0.190000  12.86  13.69  3.832e+02 -7.804e+01 -7.804e+01 -1.483e+03  1.004e+04
5.313e-01  1.559e-11
 26  0.210000  14.03  14.02  3.010e+02 -4.669e+01 -4.669e+01 -9.805e+02  1.021e+04
5.819e-01  1.002e-11
 27  0.230000  15.03  14.20  2.467e+02 -2.608e+01 -2.608e+01 -5.998e+02  1.035e+04
6.231e-01  6.991e-12
 28  0.250000  15.97  14.38  2.079e+02 -1.280e+01 -1.280e+01 -3.201e+02  1.048e+04
6.616e-01  5.085e-12
 29  0.270000  16.84  14.57  1.789e+02 -4.082e+00 -4.082e+00 -1.102e+02  1.060e+04
6.985e-01  3.811e-12
 30  0.290000  17.66  14.64  1.565e+02  1.811e+00  1.811e+00  5.251e+01  1.071e+04
7.285e-01  2.989e-12
 31  0.310000  18.45  14.76  1.386e+02  6.201e+00  6.202e+00  1.922e+02  1.081e+04
7.588e-01  2.376e-12
```

Following this are the inner boxes along the Z axis, then the same format showing the model along the Y and X axes.

That completes the definition of the model. We have not yet defined the model grid in setup.c

setup.c

This file, setup.c, is an include file for cmain.c. It is C code, but without the beginning and end of a function definition. Each time setup.c is modified, the code needs to be re-compiled with the Makefile. This is very fast.

The comments in setup.c explain most of the options. A few words of background are required for some options.

The number of lines and levels, n_lines and n_states. For simple rotors such as CO, the number of quantum states is always one more than the number of lines, for example, 3 states would include J=0,1,2 and the 2 lines would include J=(2-1) and J=(1-0). Any number of states and lines can be calculated. If the number of states is very high and the gas temperature very low, the upper states will have no significant population, just numerical noise. This can lead to some odd behavior. MOLLIE usually detects the problem and complains with warnings to stdout and Cnotes0.

The number of states and lines is defined in 3 places, setup.c, nlines_f77.h, and nlines_C.h, and they must match. If not, the MOLLIE should detect the differences and not run. The nlines files are include files for the C and fortran programs. Because the code is compiled for each molecule, the information on the molecular lines needs to be available at compilation. (This could be done with more complicated configure and make files.)

For more complicated molecules, there are specific numbers of states and lines. These are listed in the comments in setup.c and also in the file nlines_f77.h.

There are 2 different methods to model N2H+: first, the approximation of hyperfine statistical equilibrium (HSE); the second, a non-LTE calculation of the individual hyperfine lines. The two approximations are explained in Keto and Rybicki 2011.

Using the HSE approximation, molecule=N2Hplus. The set up for N2H+ is the same as for CO in that the number of lines is always one less than the number of states. To model the (1-0) and (3-2) transitions, set the input number of states to n_states=4, the number of lines to n_lines=3, and the number of output lines to number_lines=2, and the list of output lines to linelist[0]=0 and linelist[1]=2. The variable number_lines is the dimension of linelist, and linelist lists which lines are included in the output ModelCube0.

To model N2H+ with non-LTE hyperfine, molecule=N2Hhyp. The number of lines and states must be one of several combinations. These are listed as comments in setup.c. The number of rotational states can be determined from n_lines. For example, n_states=10, n_lines=1, means that the 10 hyperfine states of the J=0 and J=1 rotational states are modeled, and 1 spectral line will be calculated, the J=1-0 transition including the 7 hyperfine lines.

Execution

After editing setup.c, define_model.c, and nlines_f77.h and nlines_C.h run the Makefile with make. There are 2 example Makefiles. The one named Makefile compiles the code on my laptop using the mpicc and mpif77 commands that were installed by the system package manager. The one named Makefile_cfa uses a version of mpich2 installed in my own directories. A significant difference is that the newer gcc and gfortran require that the code be linked by gfortran (FCC in the Makefile) as shown in the Makefile. In the older version for my CfA installation, the code is linked by gcc and requires the location of Fortran SO libraries for I/O as shown in the Makefile_cfa

The compilers mpicc and mpif77 are defined in the installation of MPI. You cannot use gcc and g77 here. For example, gcc and g77 (or gfortran) are used to compile the MPI installation which then defines mpicc and mpif77. Alternatively, you could use the faster Intel compilers to install MPI, then mpicc and mpif77 would be based on the Intel compilers. A cluster will have various options available.

Setup up MPI by starting the mpd daemons if necessary or defining the hosts. This is very specific to your installation. A cluster will require other definitions.

Finally, something like,

mpiexec –n 65 ./c.x

will execute MOLLIE as an interactive program with 65 processors. If you are using a laptop or desktop with a few processors, use one more process than there are processors. For example, if you have 4 processors, then

mpiexec –n 5 ./c.x

The reason is that the master and slaves do not run at the same time. So the first process, the master is idle while the 4 slaves are computing.

If you have installed MPICH2 and gfortran on your own linux machine and you are using only the processors on that machine, the command above is all that is required. To use multiple machines, you need to define the hosts. This is specific to your installation.

Alternatively, you may need to submit the job through the queuing for your cluster.

Keep in mind that MOLLIE always writes to ModelCube0, Cnotes0, etc. The names do not change. If MOLLIE runs to completion, any old files are overwritten. If MOLLIE detects a setup error and exits gracefully without completing the radiative transfer calculation, then any old files remain as they were. If you are not paying

attention, it is possible to assume the code ran correctly, then go on to read an old ModelCube0 left over from a previous run.

It is always worth checking the end of Cnotes0 for the lines indicating successful completion,

```
P0: Elapsed time to complete model 0 was  1985 secs
P0: Total elapsed time  33.427283 minutes
```

Successful completion is also indicated by the printed output on the terminal if MOLLIE is run interactively or in the log file if run through a queuing system on a cluster. Successful looks like,

```
File ./ModelCube0 opened
wrote 307435 data to cube
wrote 1229676 bytes to cube
Output file written
P0: Elapsed time to complete model 0 was  4 secs
P0: Total elapsed time  0.101583 minutes
P0: Sent final call for slaves
P0: Process complete, exiting now ...
```

Reading the Output

The output, aside from the Cnotes0 file, is a binary file ModelCube0 that is read with the IDL program load_model.pro. These IDL programs also work with the open source GDL (gnu data language not to be confused with gnome development language). Gnu data language is available as a package for Debian based linux (Ubuntu), but there is no RPM yet for the other linux.

Included in the directory are IDL files,

idl_setup
rt_startup
rt_init.pro
my_plot.pro
load_model.pro
fits.pro
fits2.pro
rt_commons
myconv.pro
conv2d.pro

To start IDL, source idl_setup which makes all the data accessible from the command line through the common blocks. Edit this file first to match the installation of IDL on your computer.

```
source idl_setup
idl

IDL Version 7.0 (linux x86_64 m64). (c) 2007, ITT Visual
Information Solutions
Installation number: 100554.
Licensed for use by: Smithsonian Astrophysical Observatory

% Compiled module: RT_INIT.
rt_startup completed
```

Now read in ModlCube0

```
IDL> load_model


% Compiled module: LOAD_MODEL.
Opening file ModelCube0
nlines as read            2
swapping bytes ?  no
there are            2 lines in this data set
there are            1 views in this data set
the numbers of channels are          373         403
grid size nx ny           64          64
cell size    0.00500000   0.00500000
beam size    0.00835530   0.00835530
The lines are:
N2H+(1-0)       93.173703  GHz
N2H+(3-2)       279.51170  GHz
velocities for line N2H+(1-0)     -8.95061      7.47244
velocities for line N2H+(3-2)     -3.99279      4.04721
longitudes
      0.00000
latitudes
      0.00000
max in line        0 is       3.22367  at v,i,j        0
31      31
max in line        1 is       1.92465  at v,i,j        0
31      31
NLINES (CUBE)   LONG      =            2
NVIEWS (CUBE)   LONG      =            1
NX (CUBE)       LONG      =           64
NY (CUBE)       LONG      =           64
NCHAN (CUBE)    LONG      = Array[2]
```

```
minimum and maximum brightness in data set  -5.84931e-09
3.22367
```

The data is stored in the 5 dimensional array data, brightness as a function of:

Line
View
X coord
Y coord
Velocity

The 2 dimensional array chvel stores the channel velocity for the data. For spectra with hyperfines such as N2H+, the data cube skips over velocities with no brightness. You need to have chvel on the X axis of the spectrum to get the spacing between the hyperfines correct.

This plots the spectrum in the center of the 64x64 map.

```
IDL> plot,chvel[0,*],data[0,0,32,32,*]
```

The simplicity of the command above is very useful for debugging problems. This plots the data with no other programs involved.

If the data has not been convolved with a beam, then the model will not match an observation. To convolve the data in IDL, run the program myconv.pro. First edit this program to change the FWHM to match your observation.

To write a FITS file, fits.pro is a one line program that writes a FITS file with minimal header information. fits2.pro writes a VLA style header including most of the header data for fits readers such as DS9 to work properly. Edit fits2.pro to change the header information to match your observation. Both of these programs require the GSFC FITSIO package for IDL.

To run the fits program, first select your line and viewing angle. Assuming that there is one viewing angle, view=0 is the default,

```
line = 0

IDL> fits2
JYPIXELCONV     DOUBLE    = Array[2]
DATAOUT         DOUBLE    = Array[64, 64, 403]
SIMPLE  =                    T / Written by IDL:  Fri Jun
22 23:07:12 2012
BITPIX  =                  -64 / Number of bits per data
pixel
NAXIS   =                    3 / Number of data axes
NAXIS1  =                   64 /
```

```
NAXIS2  =                   64 /
NAXIS3  =                  403 /
DATE    = '2012-06-23'         / Creation UTC (CCCC-MM-DD)
date of FITS header
COMMENT FITS (Flexible Image Transport System) format is
defined in 'Astronomy
COMMENT and Astrophysics', volume 376, page 359; bibcode
2001A&A...376..359H
OBJECT  = 'L692-4  '           /
TELESCOP= 'MOLLIE  '           /
INSTRUME= 'MOLLIE  '           /
OBSERVER= 'MOLLIE  '           /
DATE-OBS= '2012-06-22'         /
DATE-MAP= '2012-06-22'         /
BSCALE  =              1.00000 /
BZERO   =              0.00000 /
BUNIT   = 'JY/PIXEL'           /
EPOCH   = '2000    '           /
RESTFREQ=        93173702656.0 /
CTYPE1  = 'RA---SIN'           /
CRVAL1  =        295.268750000 /
CDELT1  =           0.00114592 /
CRPIX1  =                   32 /
CROTA1  =              0.00000 /
CTYPE2  = 'DEC--SIN'           /
CRVAL2  =        10.9505555556 /
CDELT2  =           0.00114592 /
CRPIX2  =                   33 /
CROTA2  =              0.00000 /
CTYPE3  = 'FREQ    '           /
CRVAL3  =        93173696444.4 /
CDELT3  =        6211.58017707 /
CRPIX3  =                    1 /
CROTA3  =              0.00000 /
END
```

The output is called “fits_file”. An example is included in the directory.