Torus User Manual

22 Mar 2017

Contributors:
Alicia Aarnio, David Acreman, Ben Ambery Smith, Josh Hamilton, Tim Harries, Thomas Haworth, David Rundle.
TWiki to PDF Converter TWiki2pdf Version 2.3.4 by Steve Lloyd.

Contents

Preface

Exeter is home to the 3-dimensional radiative-transfer code, TORUS, which is either an acronym for Transport of Radiation Under Sobolev, or Transport of Radiation Using Stokes.

Chapter 1

Introduction

TORUS is a radiation transfer and radiation-hydrodynamics code developed by Tim Harries and co-workers. Originally developed at St. Andrews and UCL the code developers are now primarily based at the University of Exeter in the sunny southwest of the UK near the English Riviera[1].

The basic philosophy of TORUS is one of flexibility. The code has a basic infrastructure that includes the AMR mesh scheme (first developed by Neil Symington) that is used by several physics modules including atomic line transfer in a moving medium (developed by Tim Harries, Ryuichi Kurosawa and Neil Symington), molecular line transfer (Dave Rundle and Tim Harries), photoionization (Tim Harries), radiation hydrodynamics (Tim Harries, Dave Acreman, Tom Haworth) and radiative equilibrium (Tim Harries, Dave Acreman, Ryuichi Kurosawa). The code has been used to tackle a wide variety of problems from magnetospheric accretion onto T Tauri stars, spiral nebulae around Wolf-Rayet stars, discs around Herbig AeBe stars, structured winds of O supergiants and Raman-scattered line formation in symbiotic binaries, and dust emission and molecular line formation in star forming clusters.

As of today (3rd June 2014) the code consists of approximately 180000 lines of Fortran 2003. The code itself is written to fairly tight coding guidelines, and should be fairly readable - although it is admittedly sparsely commented we use self-explanatory variable and subroutine names as much as possible. The code also makes extensive use of fortran modules, and is compiled using a standard Gnu makefile. The code is parallelized using both MPI and OMP, and can uses these parallel sections either separately or in a hybrid mode. Most physics modules have quite low communication overhead.

TORUS has been compiled on a wide variety of system architectures. We regularly run the code at Exeter using `ifort` and compile the code nightly using `gfortran` and `nagfor`. A nightly test suite (developed by Dave A) is run from the subversion repository ensuring that the current repository version at least compiles and runs basic benchmarks for radiative equilibrium, molecular statistical equilibrium and hydrodynamics (a Sod shock tube test).

We rely very little on external libraries. If you want to have your output files as FITS (which we recommend) then you need to have the cfitsio library installed and in your linking path when compiling. You need MPI libraries (we use the intel one or openMPI here) if you want to use the MPI parallelization, and you need an OMP-directive compatible fortran compiler to access the OMP or hybrid modes. Visualization of the torus models can be achieved via any suitable 3rd party program that can read the VTK format files that TORUS produces (we prefer visit here, but paraview can be good too).

If you are reading this then you have been given access to TORUS - please respect our intellectual property and do not distribute the code. We are of course really happy for groups to use the code and are excited to see results from it appearing in the refereed literature.

Finally the TORUS code is the result of the intellectual effort of a small group of dedicated astronomers, most of whom are at a fairly early career stage. It is therefore extremely important that they get credit for their hard work!!

Tim Harries

6th January 2011

[1] *May I ask what you expected to see out of a Torquay hotel bedroom window? Sydney Opera House, perhaps? The Hanging Gardens of Babylon? Herds of wildebeest sweeping majestically...?* Basil Fawlty

Complete: ☐
Responsible:
Author: David Acreman 13 Feb 2009
Contributors: David Acreman
Last significantly modified by: David Acreman 13 Feb 2009
Not yet reviewed

Chapter 2

Building Torus

2.0.1 Getting the Torus code

The Torus code is managed using SVN. To download the code from the repository type:

where USERNAME is your username on `mail.astro.ex.ac.uk`.

2.0.2 Compilers

Torus is known to build using the freely available gfortran compiler and the commercial Intel Fortran compiler. These compilers are routinely used to run Torus so you can expect them to run typical configurations successfully. g95 has been removed from the Torus test suite as the compiler is no longer being developed and building with gfortran gives better performance.

2.0.3 Building Torus

The Torus repository contains a Makefile which can be used to build the code for a number of different systems. There are generic systems, which can be used to build Torus for a particular compiler, and more specialised systems which are used to build Torus on a particular machine (typically an HPC system). System specific options (e.g. compiler flags) are specified in the Makefile so you should not need to set these up, unless you are using a new compiler or you want to tailor the build for your specific machine.

The environment variable `SYSTEM` is used to tell the Makefile which configuration to use. For example to use the `gfortran` configuration (suitable for the gfortran compiler on Mac OS X and Linux) e.g. in csh type

```
setenv SYSTEM gfortran
```

Alternatively you can set this environment variable in your `.cshrc` file.

The this table describes the supported SYSTEM types relating to specific compilers:

SYSTEM	Description	Can use mpi=yes option?
gfortran	The GNU gfortran compiler (gcc.gnu.org/fortran)	Yes (can also use ompiosx system)
ifort	The intel compiler	Yes

The this table describes the currently supported SYSTEM types relating to specific machines:

SYSTEM	Description
complexity	The Leicester Dirac system AKA complexity cluster
zen	intel MPI compiler on the SGI-Altix system at Exeter
zensingle	single processor compilation on SGI-Altix system

You can now go into the torus directory and build the code

```
cd torus
make depends
make
```

Several options can be passed to the make process. Firstly there are options controlling the parallelisation:

Option	Description	Default	Notes
mpi	Build with MPI parallelism	Depends on SYSTEM	
openmp	Build with OpenMP parallelism	no	

The following options include or exclude major functionality from Torus. Excluding parts of the code will reduce compile time and may reduce memory use (e.g. Building without the photoion or sph options excludes some statically allocated octal components.

Option	Description	Default	Notes
hydro	Include hydrodynamics	yes	
photoion	Include photoionisation	yes	
molecular	Include molecular physics	yes	
sph	Include code for making grids from SPH	yes	
atomic	Include atomic physics	yes	

These options govern linking to external libraries:

Option	Description	Default	Notes
cfitsio	Link with cfitsio libraries	yes	
hdf	Build with hdf5 support required for reading flash dumps	no	
static	Perform static linking	no	Not available on Mac OS X

These options are useful when developing code:

Option	Description	Default	Notes
debug	Switch on debugging flags	no	
profile	Enable profiling	no	
coverage	Enable coverage analysis with gcov	no	gnu compilers only
trace	Use Intel trace collector	no	Intel compilers only

For example to build torus without linking to the cfitsio libraries and with debug flags enabled use

```
make cfitsio=no debug=yes
```

You may need to remove all files from a previous build and start from scratch (for example if you have previously compiled with debug flags and need to recompile without debug flags). The Makefile has an option to clean up files from the previous build `make clean`

FITS library

By default Torus will try to link with the FITS library. Different SYSTEMs make different assumptions about where the FITS library is located, but typically the build will look in either your home directory (`/home/username/cfitsio/lib`) or a default location (e.g. `/usr/local/lib` or `/usr/lib`).

If you know where the FITS library is, but the Torus build is not finding it, try setting the `LIBRARY_PATH` environment variable to the directory which contains `libcfitsio.a`.

If you can't find a `libcfitsio.a` file then you can build your own as follows:

1 Download the source code: `ftp ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio_latest.tar.gz`
2 Unpack the source code: `tar zxvf cfitsio_latest.tar.gz`
3 Move into the directory which contains the cfitsio source code: `cd cfitsio`
4 Configure the build: `./configure --prefix`$HOME/cfitsio=. On a Mac use `./configure --prefix`$HOME/cfit
CC=cc=
5 Build the library: `make`
6 Install the library: `make install`

2.0.4 Testing your build

Assuming that you complete compilation without error messages, you can check that your torus executable is doing the right thing using the benchmark tests that torus runs on an nightly basis. The parameter files and scripts to check the test results are located at

```
/torus/benchmarks/
```

The tests themselves cover different aspects of the code including hydrodynamics (/torus/benchmarks/hydro1d), photoionization(HII_region, HII_regionMPI) and molecular line transport (molebench). Copy your torus executable into a benchmark folder, run the code and check the results.

Complete:
Responsible:
Author: David Acreman 13 Feb 2009
Contributors: David Acreman
Last significantly modified by: David Acreman 13 Feb 2009
Not yet reviewed

Chapter 3

Running Torus

3.0.1 Running torus

The torus distribution contains a variety of data files such as model atmospheres, grain refractive indices and stellar evolution tracks. These are stored in the "data" subdirectory of the torus distribution. In order for the code to access these the environment variables `TORUS_DATA` must be set e.g. in csh and tcsh

```
setenv TORUS_DATA /mypathtotorus/torus/data
```

for bash and similar shells use

```
export TORUS_DATA/mypathtotorus/torus/data=
```

Once you have compiled the code you will have an executable called `torus.xxx` where `xxx` is defined by your `SYSTEM` type (e.g. `torus.gfortran`). In order to run the code you need to execute the torus executable in a directory that contains a parameters (input) file. For example if you have a directory /home/myusername/model then `cd` into that directory and run torus:

```
$ /my-path-to-torus/torus.xxx
```

Torus will then run, and will look for a file called parameters.dat. If you have called your parameters file something different (e.g. `mymodel.dat`) then

```
$ /my-path-to-torus/torus.xxx mymodel.dat
```

will run torus on mymodel.dat

Complete:
Responsible:
Author: Tim Harries 24 Aug 2010
Contributors: Tim Harries
Last significantly modified by: Tim Harries 24 Aug 2010
Not yet reviewed

3.0.2 Creating a parameters file

The parameters file basically consists of a list of keywords and associated values. The keywords are case sensitive, and should appear as the first non-whitespace characters on a given line of the input file. The next non-whitespace characters should be the value associated with that keyword. Characters after the keyword/value pair are ignored, and lines that start with a # or ! keyword are also ignored. Keywords may appear in any order in the input file (although for clarity you will wish to collect keywords referring to particular part of the model together). This gives significant flexibility in formatting the TORUS parameter file for maximum clarity. *Note that each unique keyword should appear only once in the parameters file. Also note not to use 'Tab' for spacing when creating a parameters file.*

As an example say we have a keyword **nphotons** which we wish to assign a value of 1000. We could write this as

```
nphotons 1000
```

as one line in the parameters file. But this doesn't contain any comments to remind the user what this is for, so it may be better to write

```
nphotons 1000 !  The number of photons used to produce the SED
```

The above is more user-friendly. We can also put comments on their own lines, but make sure they start with a # or a ! e.g.

```
! The following parameters describe the image
nphotons 1000 ! The number of photons used to produce the SED
imagesize 100  ! The linear image size in AU
npixels 100   ! Produce a 100x100 pixel image
noscat   T      ! calculate image without scattered light
```

The values associated with keywords may be integer, logical (T or F), or floating point. Normally only one value is associated with each keyword, although a limited number require an array of values (separated by whitespace) e.g.

```
sourceposition1 0.  12.  7.  !  first source position in units of 10^10 cm
```

Only a subset of keywords must be specified at run-time. Others will take on a default value which is *normally* sensible. If a keyword is not in the parameters file but it is required for the model the code will exit and tell you that a particular keyword/value pair must be specified.

TORUS will stop with a fatal error if a particular keyword appears more than once in an parameters file. It will also report keywords that were unused on input. To check your parameters file run TORUS with the word "check" afterwards. This causes TORUS to read your parameters file then exit, so that you can test for errors without actually running the model.

Here is an example TORUS input parameter file for the Pascucci circumstellar disc benchmark.

```
! Torus v2 parameter file for 2D benchmark disc
! See Pascucci et al, 2004, A&A, 417, 793

dustphysics T   ! use dust microphysics

radeq T  ! perform a radiative equilibrium calculation

! AMR grid parameters

readgrid F   ! we aren't reading a grid, we will set one up from scratch
writegrid F  ! we don't need to write out the AMR file - we just need SEDs
amrgridsize 4.e6 ! the linear size of the top-level AMR mesh in units of 10^10
   cm
```

```
amr2d T  ! this is a 2d (cylindical) model

! grid smoothing switches

smoothgridtau T  ! smooths the grid for optical depth, in order to resolve disc
    photosphere
dosmoothgrid T   ! smooth the grid for jumps in cell refinement
smoothfactor 3.0 ! make sure that neighbouring cells are not only one AMR depth
    apart


! Source parameters

nsource 1  ! there is just one source
radius1 1.  ! it has a radius of 1 solar radius
teff1 5800.  ! the source effective temperature
contflux1 blackbody ! the continuum flux is assumed to be a blackbody
mass1 1.   ! the source has a mass of one solar mass
sourcepos1 0. 0. 0  ! it is located at the grid centre

! Geometry specific parameters

geometry benchmark  ! this is the Pascucci (2004) benchmark
rinner 1. ! inner disc radius (AU)
router 1000. ! outer disc radius
height 125. ! disc scaleheight at 100 AU (in AU)
rho 8.16136e-18  ! density at inner edge midplane (g/cc)

! Dust grain properties

iso_scatter T      ! Assume isotropic scattering (assumed by benchmark)
graintype1 sil_dl  ! Drain and Lee silicates
amin1 0.12         ! minimum grain size (microns)
amax1 0.1201       ! maximum grain size (microns)
qdist1 0.01        ! power law index (flat)

! Output SEDs

spectrum T   ! produce a spectrum

! SED parameters

ninc 2                ! number of inclinations
firstinc 12.5         ! the first inclination (degrees)
lastinc 77.5          ! the last inclination (degrees)
nphotons 500000       ! the number of photon packets in each SED
filename test         ! the root of the output filename
sised T               ! Write spectrum as lambda vs F lambda in SI units
distance 2.25558e-8   ! Distance to observer
sedlammin 0.12        ! minimum wavelength in SED file
sedlammax 2000        ! maximum wavelength in SED file
sedwavlin F           ! Linear spacing in SED file?
```

3.0.3 Top level keywords

The basic architecture of the parameters file follows three stages. The first is to setup up the AMR mesh, the second is to perform some physics (radiative equilibrium, hydrodynamics etc) and the third is to produce some outputs (dust continuum images, molecular line datacubes, Halpha spectra etc). Here we describe the most important keywords for setting up the model. Naturally each choice of grid setup, physics, and outputs, leads to a further set of parameter keywords that must be specified, which are detailed in the chapters that deal with the individual physical models that can be computed.

3.0.4 Setting up the AMR grid

There is a choice of AMR dimensionality and coordinate system. This is done by setting one (and only one) of `amr1d` or `amr2d` or `amr3d` to `T`. The standard 3-d cartesian grid is specified by

```
amr3d T ! grid is three dimensional
```

and gives a grid of 2x2x2 cubes each of which can be further sub-divided, leading to an AMR structure stored as an octree. Should your problem contain a high degree of rotational symmetry then you may wish to employ the 3-d cylindrical grid, specified as follows:

```
amr3d T ! grid is three dimensional
cylindrical T ! use cylindrical coordinates
```

In the cylindrical system x is radius, z is height and the third dimension (phi) is azimuthal angle. Each cell in the grid can have either four children (the cell is split in x and z but not phi) or eight children (cell is split in x, z, and azimuthal extent).

Should your model be axisymmetric (such as a disc) the grid should be specified as 2d, using

```
amr2d T ! grid is 2-d cylindrical
```

which gives a quad-tree grid where x is the radius and z is height. (Note that in the hydrodynamical case only the 2d geometry is a cartesian surface).

One dimensional grids are specified by

```
amr1d T ! grid is one dimensional
```

This provides an AMR bi-tree, where the x-coordinate becomes radius. This is suitable for spherically symmetric models (the molecular benchmark is an example of this type of mesh. (Note that in the hydrodynamics case only the amr1d mode has x as a linear coordinate, not a radius.)

The size of the AMR mesh is give by `amrgridsize` which refers to the full extent of the root cell. The distance is given in units of 10^{10} cm, which are the general code units for distance. The centre of the AMR mesh is assumed to be at (0, 0, 0) for the three-d cases and (r=amrgridsize/2, z=0) for the 2-d cylindrical mode. The centre of the grid can be chosen by specifying one or more of `amrgridcentrex`, `amrgridcentrey` and `amrgridcentrez`.

The cell depth in TORUS is measured from 1 for the root cell to N where amrgridsize/(2^{N}) is the dimension of the smallest cell in the AMR mesh . The grid is always split to a depth specified by `mindepthamr` (default 5) and the maximum depth is set using `maxdepthamr` (which defaults to 31). Note that the memory footprint of the AMR mesh is strongly dependent on these two parameters, and these should be adjusted with caution (and `maxdepthamr` should certainly *not* be revised upwards, only downwards).

3.0.5 Geometry

The `geometry` keyword specifies the type of model you wish to run, be it a protostellar disc, a stellar wind, a Bonnor-Ebert sphere, etc etc.

The following non-exhaustive list provides the most commonly used geometries in TORUS:

Geometry	Description
molbench	The molecular benchmark test of xxxx
benchmark	The Pascucci (2004) disc benchmark
shakara	An alpha-disc
ttauri	magnetospheric accretion model
cmfgen	Uses input opacities from Hillier's cmfgen code
cluster	Uses a cluster from Bate's SPH models as input
theGalaxy	Uses Dobb's SPH galaxy simulation as input
molcluster	Uses one of Bate's SPH simulations as input
wr104	Spiral WR nebula model
clumpyagb	A clumpy AGB wind
starburst	A starburst simulation
runaway	Uses VH1 hydro simulation as input
protobin	Protostellar binary formation model (a la Bate et al. 1995)

Specifying a geometry usually defines a set of other keywords (such as the disc scaleheight, the microturbulence etc etc) than must also be assigned values. Specific cases are dealt with later in the manual.

3.0.6 Physics

There are several different microphysical processes that may be considered. Some of these can be considered simultaneously (e.g. dust and photoionization, or dust and molecules). Some processes are mutually exclusive (for example molecular physics and atomic physics). Setting any of these keywords to true will then mean that a further subset of keywords must be specified, which are detailed later.

Physics	Description
dustphysics	This will include dust emission/absorption processes
molecularphysics	This will include molecular opacities
atomicphysics	Atomic physics (currently limited to H/He)
photoionphysics	Photoionization physics (H/He/C/N/O/S/Si/Ne etc)
supernovae	Supernovae feedback (end stage >8 solar mass sources)
stellarwinds	Stellar wind feedback (for massive stars)

3.0.7 Type of calculation

Once the geometry has been specified, and the different microphysical processes that need to be considered have been detailed, the user must select the type of calculation they wish to perform:

- Statistical equilibrium `stateq T` . Conduct a statistical equilibrium calculation by solving the rate equations. If you have `molecularphysics T` then an algorithm described in Rundle et al. (2010) is used. If `atomicphysics T` then a comoving frame algorithm is employed.
- Radiative equilibrium `radeq T`. This requires that `dustphysics T` and uses the Monte-Carlo radiative equilibrium method of Lucy (1999).

- Photoionization equilibrium `photoion eq`. Solves photoionization equilibrium including thermal equilibrium using a method similar to that of Ercolano xxxx. Dust may be included.
- Hydrodynamics `hydrodynamics T`. This solves hydrodynamics using a TVD scheme and superbee flux-limiter.

3.0.8 Checking your parameters file

Once you have written a parameters file you can get Torus to check it for you. Run the Torus executable with the word `check` afterwards, optionally followed by the name of your parameters file (if it is not called parameters.dat). For example you can use the command `torus.openmp check mypar.dat` to check a parameters file called `mypar.dat` using a torus executable called `torus.openmp`.

Complete: ☐
Responsible:
Author: Tim Harries 07 May 2010
Contributors: Tim Harries
Last significantly modified by: Tim Harries 07 May 2010
Not yet reviewed

3.0.9 Generating and using restart dumps

Torus has the capability to write a restart dump, which is a file that contains all the information on the AMR grid. This allows you to save the results of a calculation to use again later on. For example if you calculate level populations using the molecular physics module you can use these results to generate a range of different synthetic observations with subsequent runs. To tell Torus to write a restart dump set `writegrid T` in the parameters file and set `outputfile` to the name of the file to be written. If this option is selected then Torus will write a restart dump after completing the physics calculations. To tell Torus to use a restart dump set `readgrid T` in the parameters file and set `inputfile` to the name of the dump file to be read. If this option is selected then Torus will not set up the AMR grid but will use the grid stored in the dump file.

Restart dumps are also written at other points, for example after completing an iteration of the Lucy algorithm. This allows a calculation to be restarted if it is interrupted part way through. These dump files are produced even if `writegrid` is not set but they are used in the same way as described above by setting `readgrid T` in the parameters file.

If `supernovae T` is set, the dump time is automatically changed to 10E8 at the point of the first SN explosion. Therefore `tdump 10.E8` must be specified if restarting from a point post-SN.

Complete: ☐
Responsible:
Author: David Acreman 20 Sep 2013
Contributors: David Acreman
Last significantly modified by: David Acreman 20 Sep 2013
Not yet reviewed

Chapter 4

Plotting Torus Data

TORUS used to produce graphical output using internal PGPLOT calls, but these calls have now been removed in order to increase TORUS portability.

4.0.1 Plotting images

VTK files

Output of grid data for plotting may now be made using calls to vtk_mod subroutines, which produce VTK format files. VTK files produced by Torus on the fly include:

filename	**Description**
lucy.vtk	Overwritten with each Lucy iteration, contains crossings, deltat, dust1, etacont, etaline, fixedtemp, mesh_qualit
bias.vtk	Contains: chiline, mesh_quality, and temperature.
beforesmooth.vtk	
aftersmooth.vtk	Contains: inflow, mesh_quality, rho, temperature, and velocity_magnitude.
rho.vtk	Contains: inflow, mesh_quality, rho, temperature, velocity_magnitude.

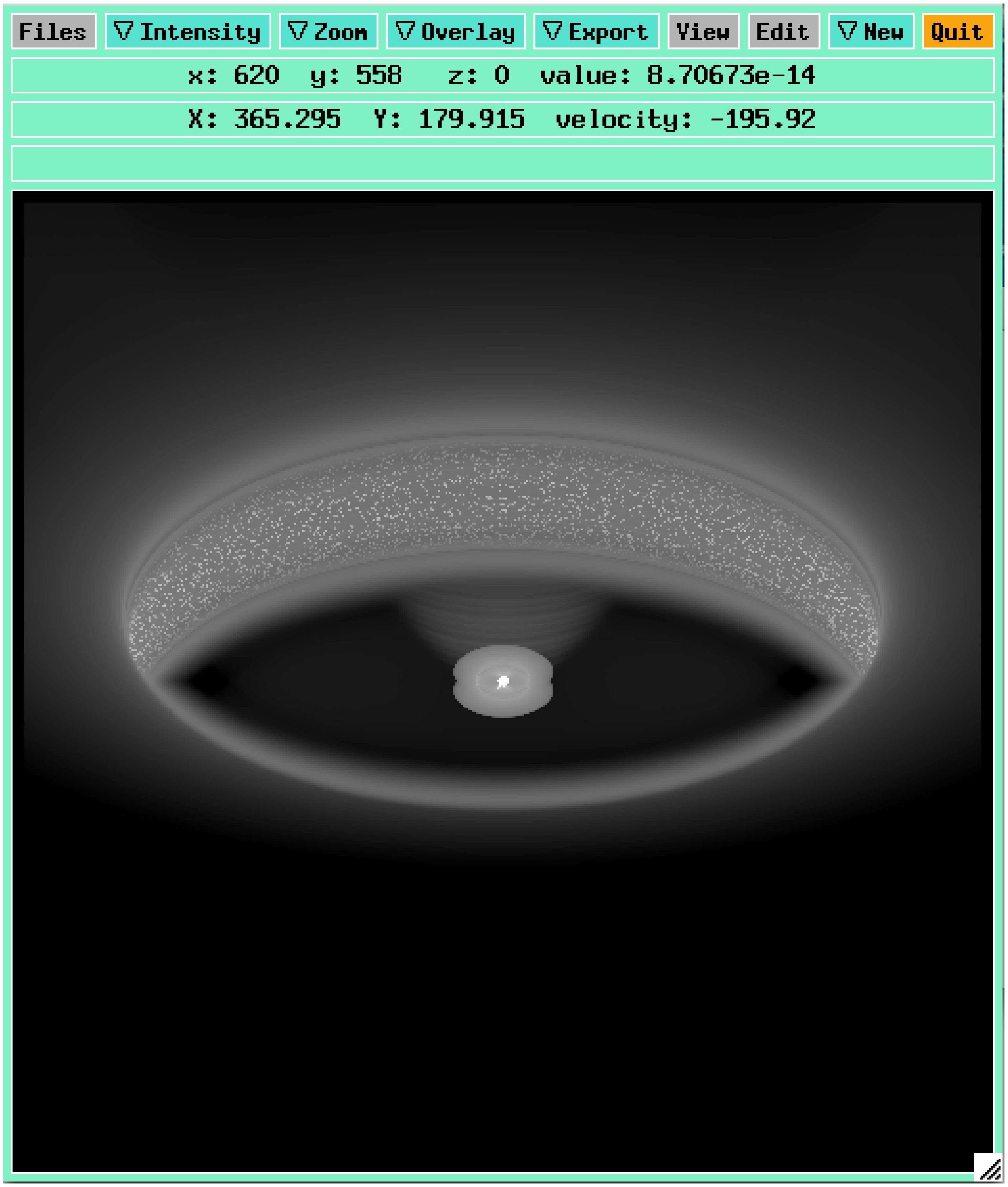
VTK files can be plotted with the VisItVisualization Tool. You may also like to try paraview.

FITS images

See the Dust Continuum Models section (subsection "Calculating Images") for more information on how to produce image files. The SAO DS9 GUI can be used to plot FITS files.

To view spectral line profiles, the program kvis handles data cubes (output as described in the atomic line transfer calculations section) effectively. The following is an example of the fits output of an atomic line transfer model run:

- KvisScreenShot1:

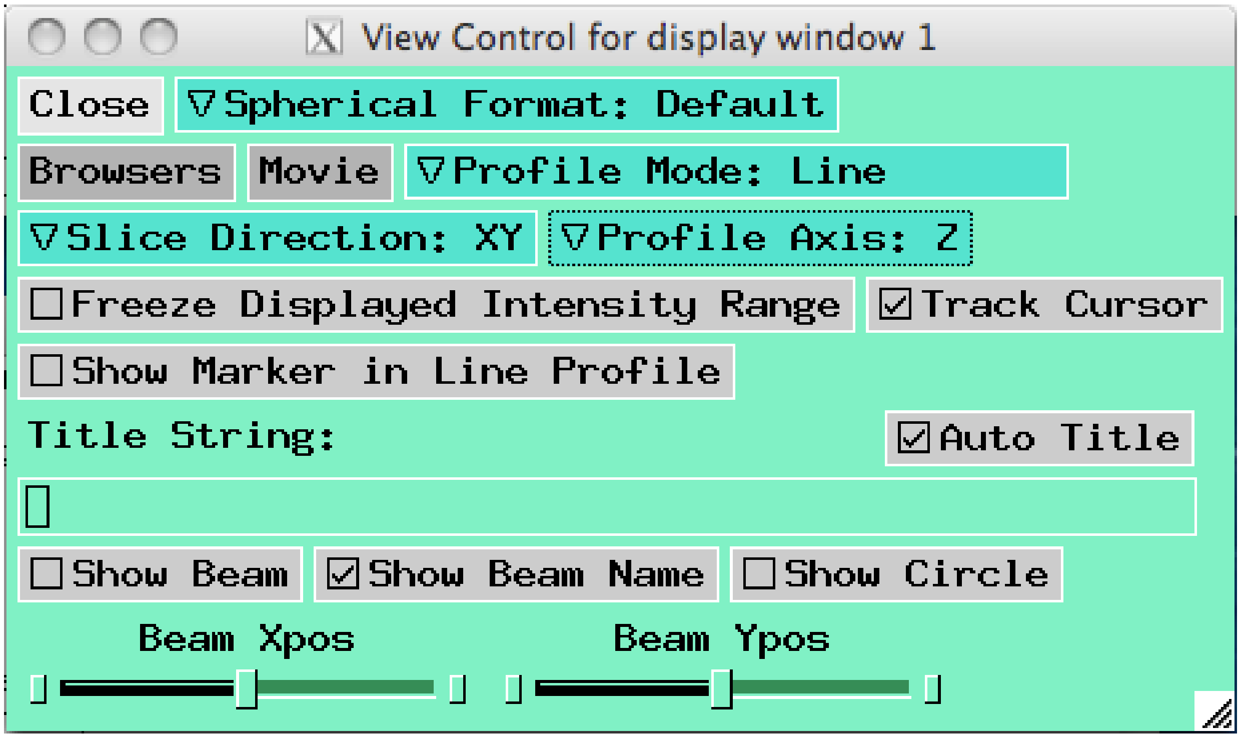


x: 620 y: 558 z: 0 value: 8.70673e-14

X: 365.295 Y: 179.915 velocity: -195.92

Clicking **View** will open this window:

- KvisScreenShot2:



View Control for display window 1

Close | ∇ Spherical Format: Default

Browsers | Movie | ∇ Profile Mode: Line

∇ Slice Direction: XY | ∇ Profile Axis: Z

☐ Freeze Displayed Intensity Range | ☑ Track Cursor

☐ Show Marker in Line Profile

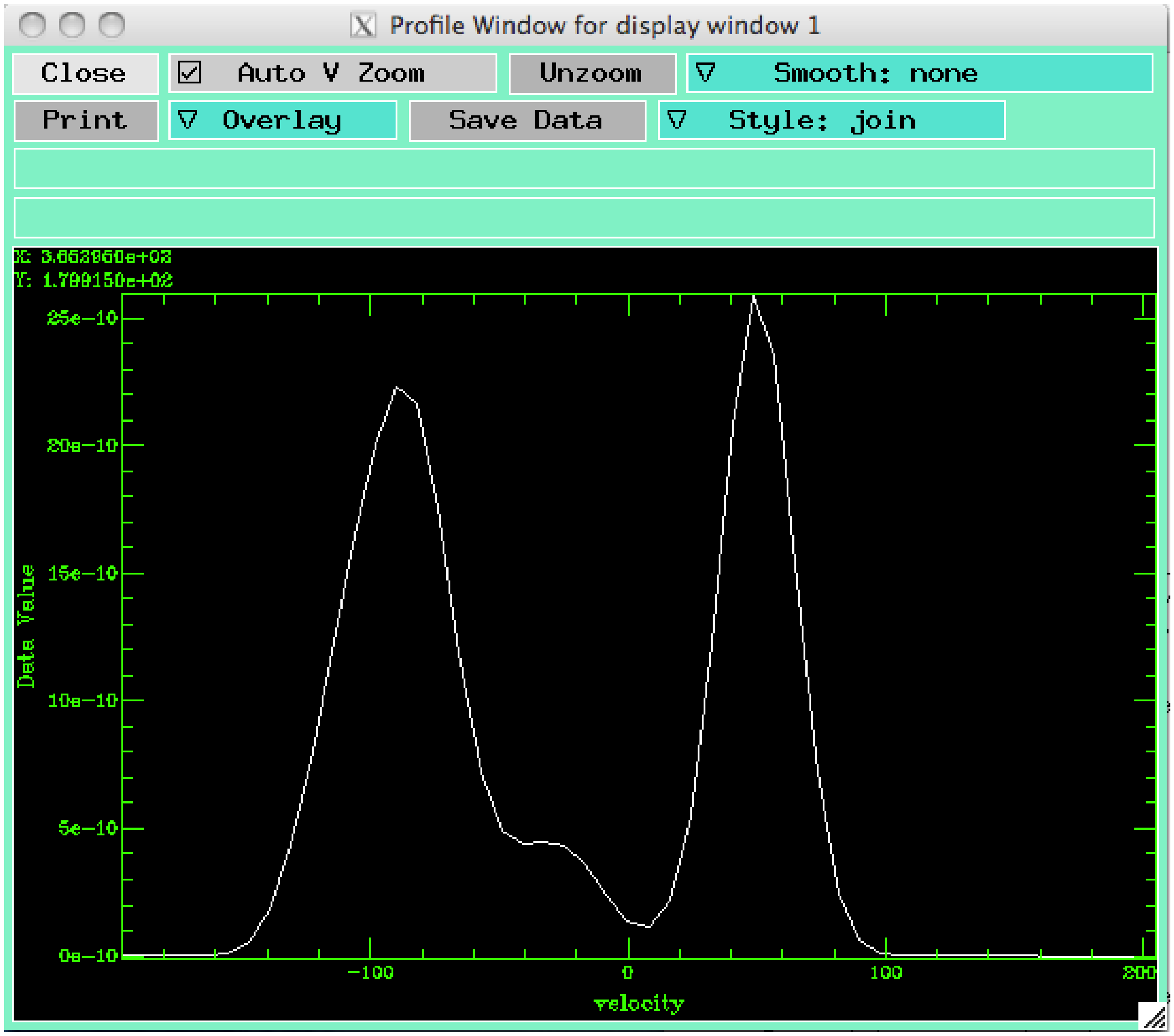
Title String: ☑ Auto Title

☐ Show Beam | ☑ Show Beam Name | ☐ Show Circle

Beam Xpos | Beam Ypos

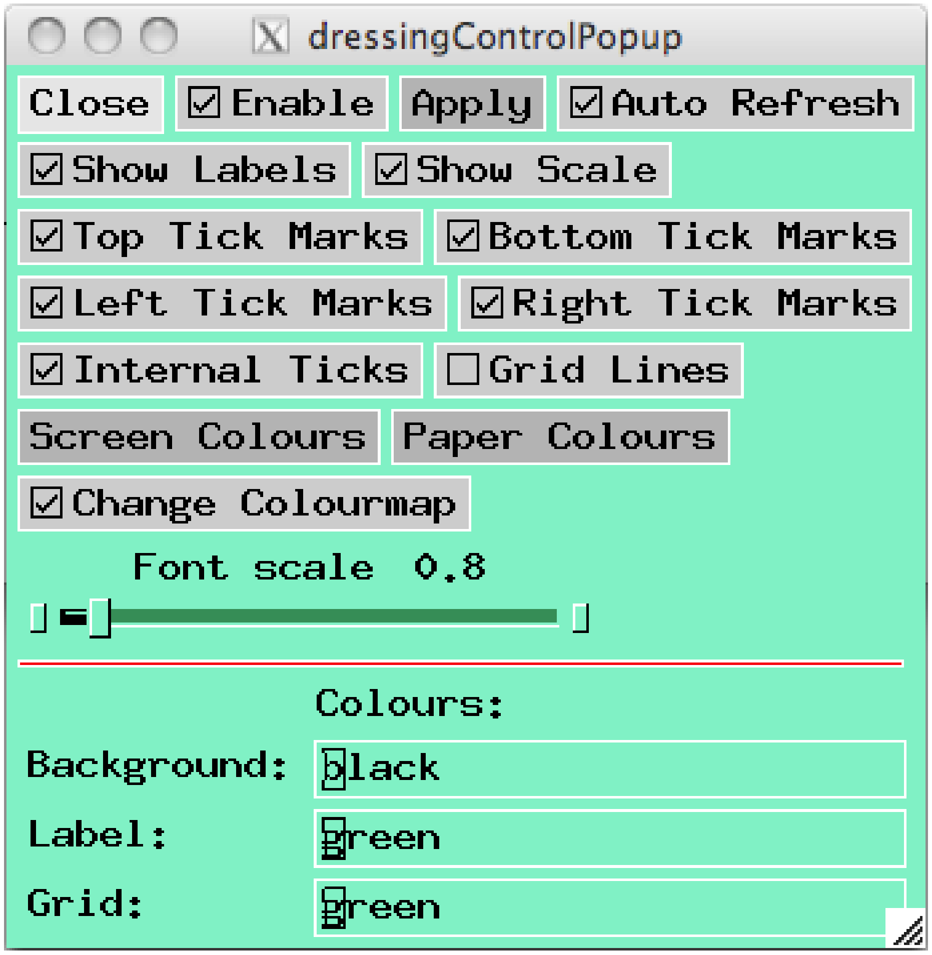
Once **Profile Mode** is changed from "none" to any other option, a profile viewing window will pop up (see next image for example). To view spectral lines wherever your cursor is, choose "line" as the profile mode, and be sure the "track cursor" box is checked. As you move the mouse around the main display window, the line profile will auto-update in the profile window.

- KvisScreenShot3:



Auto V Zoom scales the intensity axis on the fly depending on where your cursor is; uncheck this (and click **Unzoom**) to fix the y axis display to the full range. To adjust the display settings in the profile window, click **Overlay** and select "axis labels." This box will pop up:

- KvisScreenShot4:


dressingControlPopup

Close [✓ Enable] Apply [✓ Auto Refresh]

[✓ Show Labels] [✓ Show Scale]

[✓ Top Tick Marks] [✓ Bottom Tick Marks]

[✓ Left Tick Marks] [✓ Right Tick Marks]

[✓ Internal Ticks] Grid Lines

Screen Colours Paper Colours

[✓ Change Colourmap]

Font scale 0.8

Colours:

Background: black

Label: green

Grid: green

Once "enable" is checked, select display settings as desired.

Additional notes

- To generate a box average of the spectral line in a given region, change **Profile Mode** to "box average," and middle click-drag (alt+click-drag on macs) to define a rectangular region on the image. After a moment, the averaged line profile within your defined region will appear in the profile window.
- **kvis** can also generate movies- click **Movie** in the view control window. The default setting is for the X-Y plane to be the "slice direction," with the movie progressing through z values in the data cube (in our case, velocities).
- The scroll wheel on the mouse can be used to zoom in/out in the main display window, and control+left click-drag can be used to define a rectangular zoom area in the image.

4.0.2 Plotting spectral energy distributions

The creation of .dat files containing spectra is described here (subsection Calculating SEDs). SEDs can be plotted using IDL with relative ease (will attach sample code here soon).

Complete:
Responsible:
Author: Alicia Aarnio 26 Aug 2010
Contributors: Alicia Aarnio
Last significantly modified by: Alicia Aarnio 26 Aug 2010
Not yet reviewed

Chapter 5

The physics modules

5.1 Dust Continuum Models

5.1.1 Introduction

Torus uses the radiative-equilibrium method of Lucy (1999) to calculate dust temperatures for arbitrary distributions of dust illuminated by multiple radiation sources. New users should read and become familiar with the photon-packet algorithm described in Lucy's paper. The following chapter details the input keywords required to compute a radiative-equilibrium model and to calculate emergent SEDs and images. Some pitfalls and gotchas are also described which should help the novice user.

5.1.2 Describing the size and chemistry of the dust

The dust grain size distribution and chemistry must be specified using keywords if `dustphysics T` is set. An arbitrary number of different grain sizes and chemistries can be listed by the keyword `ndustype`, which requires an integer value. The minimum grain size (in microns) is set using `amin`, the maximum using `amax` the power-law index as `qdist`.

```
graintype1 draine_sil ! Draine (2003) silicates
grainfrac1 0.01          ! 1:100 dust:to:gas mass ratio
amin1 0.1          ! minimum grain size (microns)
amax1 1.0        ! maximum grain size (microns)
qdist1 3.5         ! power law index (a^-3.5)
```

The following table gives details of the different grain chemistries currently coded:

graintype	Description
draine_sil	Draine (2003) silicate grains

Alternatively you can provide your own grain optical constants in a text file. This file must have three space-delimited columns (Wavelength (microns), Real part of n, imaginary part of n). Comments starting in the first column with a hash symbol are ignored. You must also specify the density of the grain material in grams per cc.

```
graintype1 mygrains.dat
graindensity1 3.5 ! density of grain material in grams/cc
grainfrac1 0.01          ! 1:100 dust:to:gas mass ratio
amin1 0.1          ! minimum grain size (microns)
amax1 1.0        ! maximum grain size (microns)
qdist1 3.5         ! power law index (a^-3.5)
```

5.1.3 Setting up the initial conditions for the Lucy iterations

There are several keywords associated with the numerics of the Lucy algorithm that can be tweaked, but for which the default values will usually give satisfactory results. The number of photon packets is by default set to be 10 times the number of grid cells (in order to provide adequate sampling of the radiation field in each cell), but this can be adjusted using the `nlucy` keyword.

Solving radiative equilibrium satisfactorily requires adequate spatial resolution, and this is particularly important for the transition between optically thin and optically thick regions (such as the surface of a disc or the inner radius of a dust shell). If `smoothgridtau` is set to true in the input file an iterative sweep is performed over the grid comparing the optical depth across a given cell (at a given wavelength) with those of its neighbours. If one of the cells has an optical depth of greater than `taumax` and its neighbour has an optical depth of less than `taumin` then the more optically thick cell is split. The sweep is performed at wavelengths between `lambdasmooth` and the maximum wavelength under consideration (normally 2 millimetres). The `dosmoothgrid` parameter may also be set. This invokes a smooth that ensures that the size ratio of neighbouring cells does not exceed `smoothfactor` (by default 3). Large differences in cell AMR depths for adjacent cells can lead to numerical problems, so it is a good idea to have `dosmoothgrid T` in your input file.

5.1.4 On convergence

Lucy's method is an iterative one, that will gradually converge towards radiative equilibrium. Since the method is a Monte-Carlo one the temperatures of cells will have noise associated with them, so convergence must be defined in a statistical sense. It should be remembered that the more photon packets that are run, the better the temperature estimate will be, and the more likely it is that cells with propagate into optically thick regions. However for some models, such as a canonical T Tauri star disc, the optical depths to the midplane are such that the chance of an individual photon packet penetrating to the midplane are vanishingly small, and even very large numbers of photon packets will leave such cells 'undersampled'. Undersampled in this context means that the number of photons passing through a given cell is smaller than `mincrossings`, so the algorithm cannot be used to adequately estimate the mean intensity for that cell. At the end of a particular iteration there may be some volumes within the grid that contain many such undersampled cells. Torus employs the diffusion approximation in these regions, using the surrounding cells that do have well-sampled radiation fields as a boundary condition. The use of the diffusion approximation is controlled by a number of keywords that are initialised using sensible defaults: Cells with a Rosseland optical depth `taudiff` are always part of the diffusion calculation. Cells that are `diffdepth` from the effective surface of the structure (e.g. the disc) are always in the diffusion zone.

The iterative scheme proceeds as follows. The photon packets are run through the grid, and the new temperatures are calculated (either from the MC estimators or the diffusion approximation). The number of undersampled cells is found (these are cells that have been sampled by fewer than `mincrossings` photon packets and yet have a Rosseland optical depth of less than `taudiff`). If the percentage of undersampled cells is greater than `lucy_undersampled` then the number of photon packets is doubled for the next iteration. The convergence criterion considers the emissivity of the dust. If the percentage change of the emissivity is less than one percent compared to the previous iteration then the Lucy algorithm finishes. The file `lucy_convergence.dat` provides detailed information on an iteration-by-iteration basis of the convergence of the radiative equilibrium.

5.1.5 Common pitfalls

Here we list some common problems:

- If `mincrossings` is set too small then the MC estimator for the radiation-field, and therefore the temperature etc for some cells can be very noisy, and although the percentage of undersampled cells is small the error on the dust emissivity will be large and the convergence criterion will never be triggered.
- Dust sublimation. If your inner disc radius is too small then the dust temperature at the inner disc can exceed the dust sublimation temperature. In order to ensure you have a consistent inner disc temperature you should set `vardustsub T` in your model, while will invoke the routines to systematically produce the correct inner disc shape.

5.1.6 Calculating SEDs

Once the radiative-equilibrium is complete one can produce spectra across an arbitrary wavelength region at an arbitrary resolution (although fundamentally the resolution of the spectrum is limited by the resolution at which the grain refractive indices were originally computed). The SEDs can be output for a variety of units. To produce a spectrum set `spectrum T` in the input file, along with `filename`, which is the name for your SED file. Setting `sed T` will produce a lambda vs lambda * F_lambda file. The wavelength range of the SED will run from `sedlammin` to `sedlammax` with `sednumlam` points. Note that the wavelength spacing is normally logarithmic (`sedwavlin T` will produce linearly spaced wavelength points). The SED production is Monte-Carlo in nature (this allows for anistropic scattering and polarization to be computed), so one can specify the number of photon packets used `nphotons` (this number of packets is divided up evenly amongst the number of wavelength bins, so the higher your resolution the more packets you need for the same signal-to-noise).

The observer distance is set using `distance xx` where `xx` is the distance from the grid origin to the observer in units of parsecs.

For non-spherical geometries the SED will be inclination dependent and torus can produce a series of SEDs for different inclinations. The number of inclinations is specified using `ninc` and the individual inclinations can then be specified as an array

```
ninc 4    ! the number of inclination
inclinations 0. 30. 60. 90. ! the different inclinations in degrees
```

or as a range of values separated in uniform steps of cosine inclination

```
ninc 4    ! the number of inclination
inclinations 0 .. 90. ! the range of inclinations in degrees
```

Alternatively you can specify the first and last inclinations (again separated in uniform steps of cosine inclination)

```
ninc 4 ! the number of inclinations
firstinc 0. ! the first inclination
lastinc 90. ! the last inclination
```

If you want even spacing in inclination angle, rather than cos inclination, set `sedcosspaced F` in the parameters file.

The output filenames are then inclination dependent, with `_incxx` appended to `filename` where `xx` is the inclination of that model in degrees. Ancillary files are also output: stellar_direct is the spectrum produced photons that arrive at the observer directly from the stellar photons. The stellar_scattered spectrum arises solely from stellar photons that are scattered one or more times towards the observer. thermal_direct is the spectrum of photons that are emitted by the disc and arrive without scattering at the observer. thermal_scattered are photons produced by dust that have been scattered one or more times before arriving at the photosphere.

For 3D geometries the SED will depend on a second angle (in addition to the inclination). A position angle can be set using the `firstPA` and `lastPA` parameters which results in evenly sampled angles between `firstPA` and `lastPA` over `ninc` steps. To scan through position angles, whilst keeping the inclination constant, just set `firstinc` equal to `lastinc`. Non-zero position angles will result in `_PAxx` being appended to `filename` where `xx` is the position angle of that model in degrees. Alternatively you can use the `posangs` parameter to specify a list of angles or a range, in the same way that the `inclinations` parameter is used. If you are specifying both inclinations and position angles then this should be done using the same method (e.g. if you use the `inclinations` parameter then the position angles should be specified using `posangs`).

5.1.7 Calculating images

Images are calculated using the same code that produces the SEDs. First one must set `image T` in the input file. Torus can produce a sequence of images. In order to do this set `nimage xx` where `xx` is the number of images and append each keyword with the appropriate image number. The observer distance is set using `distance xx` where `xx` is the distance between the observer and the grid origin in parsecs. The name of the image file is set via the `imagefile` keyword and the wavelength of the monochromatic image is set by `lambdaimage` where the units of wavelength is angstroms. The `nphotons` keyword allows one to control the signal-to-noise of the image.

The axis units of the fits images are set using `imageaxisunits xx`, where `xx` can be au, pc, cm or arcsec, (the default is au). `imagesize` sets the size of one side of the (square) image in the same units as the image axes (the default size is the maximum linear size of the grid). `npixels` defines the number of pixels along one side of the image (i.e. the image has npixels-squared pixels in total). The size of the images is controlled by `imagesize` but can be overridden for one or more images, for example `imagesize3` changes the size of the third image. By default images are square but this can be changed using the `imageaspect` keyword. The y-axis is of size `imagesize` with `npixels` pixels, but the x-axis size and number of pixels will be scaled by `imageaspect`.

For a single image the observer's viewing angle is set by `inclination` (in degrees) and `positionangle` (once again in degrees). If the inclination and position angle are zero then the observer will be looking down the z-axis of the grid. The inclination angle moves the viewing vector in the x-z plane and the position angle rotates the viewing angle about the z-axis. Hence inclination corresponds to the polar angle in a spherical co-ordinate system and position angle corresponds to the azimuthal angle. When generating multiple images the `inclination` and `positionangle` keywords can be used to specify values which apply to all images, but they can be overridden e.g. `inclination3 45.0` sets the inclination of the third image to 45 degrees.

The centre of the image will be the origin of the Torus co-ordinate system, unless you specify otherwise. `imagecentrex` and `imagecentrey` can be used to move the centre of the image (units are Torus units i.e. 10^{10} cm). For multiple images a number can be appended to specify offsets for individual images e.g. `imagecentrex3`.

```
nimage 2

imagefile1 onemicron.fits
```

```
lambdaimage1 10000.
npixels1 256
inclination1 60.

imagefile2 twomicron.fits
lambdaimage2 20000.
npixels2 256
inclination2 60.
```

5.1.8 Output files

As well as the images and SEDs the radiative equilibrium calculation produces some diagnostic files.

- `tune.dat` This file contains timing information on various stages of the calculation, and may help you judge how long a particular model is going to take to run, and may help with code/system optimization.
- `convergence_lucy.dat` This file stores details of each iteration of the Lucy algorithm. The columns are the iteration number, the mean, maximum and minimum changes in the temperature in the grid at the end of the that iteration, the percentage of "bad" cells (those that are deemed to have an undersampled radiation field while not being in the diffusion regime), the dust emissivity in cgs units, and the same in units of the source luminosity, the maximum fractional change in temperature across all cells, and finally the number of photon packets used in that iteration. Remember the code is looking for a less than 1% change in the dust emissivity between iterations for convergence.
- `albedo.dat` The dust albedo computed for the dust mixture. The column description is given in the file header.
- `info_grid.dat` Details about the size, depth, number of cells etc in the AMR grid
- `lucy.vtk` Visualization data for the grid output at the end of each iteration.
- `lucy_grid_tmp.dat` The grid is dumped after each iteration in case the code crashes, the computer goes down, or there's a nuclear attack etc.

Complete: ☐
Responsible:
Author: Tim Harries 07 May 2010
Contributors: Tim Harries
Last significantly modified by: Tim Harries 07 May 2010
Not yet reviewed

5.2 Molecular Line Transfer Calculations

Setting `molecularphysics T` in the parameters file will allow the user to perform co-moving frame non-LTE molecular line calculations in Torus. There are several keywords that **need** to be defined: `moleculefile` defines the name of the molecule to for which level populations are to be found and/or the expected emission to be calculated for (e.g. `co.mol`)

5.2.1 Level Population Calculations

If the user wishes to calculate statistical equilibrium level populations for the molecule using the algorithm set out in Rundle (2010) (by setting `stateq T` with `molecularphysics T`) then it is recommended that the parameters file for TORUS includes sensible values for at least the following parameters:

Parameter	Type	Comment
`molAbundance`	*real*	Relative abundance of molecule : H_2. Overridden by `doChemistry`
`vturb`	*real*	Constant non-thermal turbulent velocity ()
`tolerance`	*real*	RMS Convergence level for level populations or standard error per pixel with adaptive pixel subsampling
`isinlte`	*logical*	Initialise molecular level populations assuming LTE
`setmaxlevel`	*integer*	Manually set maximum number of levels to consider during convergence

Notes:

`molAbundance` will be set to a constant by this parameter. There are no geometry specific routines for setting the abundance of any molecular species at the moment. There is a drop model for 13CO that can be accessed by adding `doChemistry T`. Some geometries have their own molecular abundance reading routines which will override this default setting.

As for `molAbundance`, the `vturb` parameter adds a constant non-thermal broadening parameter to the line profile function. Doing anything more clever than this will require an extra function.

In addition to these recommended parameters there are a number of others that can be set that turn on (off) some of the diagnostic information or convergence enhancing code that has been written to make TORUS quick and easy to use...

Parameter	Type	Comment	Range	de
`dongstep`	*logical*	Perform Ng acceleration steps at specified intervals	T/F	T
`plotlevels`	*logical*	Write VTK file of diagnostic data + level populations in *./plots*	T/F	F
`outputconvergence`	*logical*	Output files containing global relative change in level populations for each iteration	T/F	F
`quasi`	*logical*	Use Sobol quasi-random number generator to determine ray direction and frequency	T/F	F
`usedust`	*logical*	Includes continuum emission/absorption from dust	T/F	F
`densitysubsample`	*logical*	Interpolate density where possible in order to make smoother image	T/F	F

After each non-fixed ray iteration a restart dump is written out and this dump can be used to restart a level population calculation. To restart a molecular line level population calculation set the parameter `restart T` in the parameters file and also set `readgrid T` and specify the name of the dump file using the `inputfile` parameter. The name of the restart dump will be the molecule name with `_grid.grid` appended. The `restart` parameter only needs to be used when restarting a molecular line calculation and shouldn't be used when restarting other types of calculation.

5.2.2 Creating Data Cubes

Once you have calculated the level populations for the molecule you are interested in you can then use Torus to generate a synthetic observation. This will be in the form of a data cube with two spatial axes and one velocity (wavelength) axis. To tell Torus to generate a data cube put `datacube T` in your parameters file. In molecular calculations the observer is initially looking along the y-axis, with a default viewing vector (0,1,0). The viewing vector is adjusted using the parameters `rotateviewaboutx` and `rotateviewaboutz` which rotate in a clockwise rotation with respect to the axis direction. The viewing vector, and other information about the cube, are written to a file called `imageparams.dat`.

The intensity data cube can be written as intensity, flux or brightness temperature by specifying the `datacubeunits` parameter as `intensity`, `flux` or `tb`.

Parameter	Comment	Default
`datacube`	Generate a data cube	F
`datacubefile`	File name for data cube	Required
`datacubeunits`	Units for intensity data cube	intensity
`datacubeaxisunits`	Units for data cube spatial axes	1e10 cm
`itrans`	Which transition to calculate	Required
`nv`	Number of velocity channels	Required
`maxVel`	Maximum velocity in data cube	Required
`minVel`	Minimum velocity in data cube	-maxVel
`npixels`	Number of pixels	Required
`cubeaspectratio`	Axis ratio of spatial axes (x/y)	1
`imageside`	Spatial size of cube	Size of AMR grid
`rotateviewaboutx`	Rotate the observer's viewing vector about the x-axis	0.0
`rotateviewaboutz`	Rotate the observer's viewing vector about the z-axis	0.0
`densitysubsample`	Interpolate density where possible in order to make smoother image	F
`wanttau`	Write optical depth to a FITS file	F

5.2.3 ALMA

Although historically torus datacubes have worked on pretty much any software (GAIA, DS9 etc.) CASA, the software required to produce synthetic ALMA observations, is a lot more picky about fits headers (it caused TJHaworth a week or so of pain). It requires frequency spectral axes and is also picky about intensity units. I think it also requires astronomical (e.g. RA/DEC) spatial axes.

To make this easy, I've implemented a parameter (defaulting to F) that will give you casa-friendly cubes

== ALMA T ==

in your input deck will give you cubes with spectral axes of Hz, RA/DEC spatial axes and intensities in Jy/pixel. You can also specify

== RA ==

== DEC ==

to change the reference pixel location for the spatial axes. This will give you cubes that definitely work with CASA, you are welcome to try other things (e.g. if you want to try galactic coordinates) but I have no idea if CASA will actually like them.

I'll upload some notes on postprocessing torus datacubes with CASA shortly.

Complete: ☐
Responsible:
Author: David Rundle 02 Sep 2010
Contributors: David Rundle
Last significantly modified by: David Rundle 02 Sep 2010
Not yet reviewed

5.3 Atomic Line Transfer Calculations

Setting `atomicphysics T` in the parameters file will allow the user to performing co-moving frame atomic line calculations in Torus. There are two keywords that need to be defined: `natom` defines how many different atoms will be used simultaneously. Subsequently each of the atomic data files need to be defined:

```
natom 2
atom1 H.atm
atom2 HeI.atm
```

(Note that only the hydrogen atom is currently fully implemented).

Synthetic observations, in the form of data cubes, can be generated from atomic line transfer calculations. These are similar to the data cubes generated by molecular line calculations (see the Molecular Line Transfer Calculations section for more details). When generating atomic line data cubes the viewing vector is controlled by the `inclination` and `positionangle` parameters (these are different parameters to those used in generating molecular line data cubes). Some parameters work differently to the molecular line case and these are summarised below.

Parameter	Comment	Default
`lamline`	Wavelength of line emission	Required
`vturb`	Turbulent velocity (km/s)	Required
`inclination`	Inclination of viewing vector	0.0
`positionangle`	Position angle of viewing vector	0.0

Complete:
Responsible:
Author: Tim Harries 24 Aug 2010
Contributors: Tim Harries
Last significantly modified by: Tim Harries 24 Aug 2010
Not yet reviewed

5.4 Photoionization Models

5.4.1 Domain Decomposed Photoionization Models

Domain decomposed photoionization models are run using multiple processors (threads). The computational domain is split into equal sized subdomains, each of which is handled by an individual thread. An additional thread is also required to perform governing operations. At present models can be run using one of the following:

Dimensionality	**Nthreads**	**Description**
1D	3	grid split into 2 plus 1 control thread
2D	5	grid split into 4 plus 1 control thread
2D	17	grid split into 16 plus 1 control thread
3D	9	grid split into 8 plus 1 control thread
3D	65	grid split into 64 plus 1 control thread

There is an automatic check in place, so if you do not use one of the above then the model will not run.

The initial generation of photon packets is performed by the governor rank. This stores photon packets that are ready in a stack until there is a predefined number ready to send to the appropriate domain thread. The domain thread then receives the stack and starts processing photons as outlined above. In the event that a photon packet crosses the boundary between two computational domains, it is added to a "to send stack" that is owned by the domain rank. Similarly to before, once this to send stack reaches a predefined size it will be sent to the neighbouring thread. Photon packets are communicated in these stacks to reduce the number of times that threads have to communicate with one another, as this can significantly improve the computation time.

To run domain decomposed photoionization models, simply include:

```
splitovermpi T
photoionphysics T
photoioneq T
```

in the parameters file.

You will also have to ensure that you are actually trying to run the job in parallel. If you are using zen, see the zen wiki.

5.4.2 The Stack Limit

Photon packets are communicated between threads in stacks to reduce the communication overhead. By default a single stack size is used for all threads, this can be specified using:

```
stacklimit xx
```

It is also possible to use a different stacklimit for each MPI thread. To do this include:

```
customstacks T
numMPIthreads xx

stacklimit 200

stacklimarray0 100
stacklimarray21 50
stacklimarray33 50
```

where

```
numMPIthreads
```

is the number of mpi threads,

```
stacklimit
```

acts as the default for unspecified threads and

```
stacklimarray
```

followed by the thread number gives the specified value for that thread.

Any questions or problems, email haworth@astro.ex.ac.uk

An Important Note

The domain decomposed photoionization routine now uses buffered sending between threads. This requires the specification of a buffer size that is difficult to predetermine and cannot be modified within a single iteration. If the buffer size is exceeded then your job will crash and give the following in your stderr_torus file (when not in debug mode):

```
[cli_1]: aborting job:Fatal error in MPI_Bsend: Invalid buffer pointer, error
    stack:
MPI_Bsend(184).......: MPI_Bsend(buf=0x647c680, count=200, dtype=USER<struct>,
    dest=2, tag=41, MPI_COMM_WORLD) failed
MPIR_Bsend_isend(338): Insufficient space in Bsend buffer; requested 100800;
    total buffer size is 504475t
```

You will then have to manually increase the buffer size from its default of 5000 photon stacks (not packets) in the buffer. To do this include

```
bufferCap 100
```

in your parameters file, replacing 100 with whatever number of stacks you want available space for in the buffer. This issue should not arise very often, if at all.

5.4.3 Monochromatic Models (domain decomposed models only)

You can run monochromatic models by including

```
monochromatic T
```

in the parameters file. This uses photons with energy (13.6 + 10**-5)eV only.

5.4.4 On The Spot Approximation

To neglect the diffuse field (photons due to recombination events), include the following in your parameters.dat file:

```
nodiffuse T
```

it otherwise treated by default.

5.4.5 Simplified Thermal Calculation

You can implement a faster, simplified thermal calculation. This should only really be used for Hydrogen only models, since it assigns temperature by interpolating between pre-assigned "fully ionized" and "fully neutral" gas temperatures. TORUS uses 10000K and 10K for these states respectively. To use this simplified thermal calculation, include:

```
quickthermal T
```

in your parameters file.

5.4.6 Variance Reduction

You can improve your signal to noise by propagating more ionizing photons (energy $>$ 13.6eV, wavelength $<$ 912Angstrom) and weighting them accordingly to restore the correct result. This can now be automatically applied to any spectrum that you use by specifying:

```
biasToLyman T
```

in the parameters file. By default, this VR is turned off. If you want to customise the extent to which the bias takes place, simply use the following in the parameters file:

```
biasMagnitude 100.d0
```

and vary 100.d0 to whatever you want it to be. Note that biasMagnitude = 1.d0 is the same as running the non VR model.

Caution should always be exercised when using VR techniques.

If you want to check that your photons are still reproducing the correct spectrum, use:

```
binPhotons T
```

in the parameters file. This sums the number of photon packets, multiplied by their weight and divided by the total number of photon packets and dumps them to "bins.dat". By running this procedure for biasMagnitude = 1.d0 and then again with e.g. biasMagnitude = 100.d0 you can check that you are reproducing the correct "spectrum" by comparing the two output files. You should expect the non-ionizing part of the spectrum to be much noisier than the ionizing part.

For those who use photoionAMR_mod (i.e. domain decomposed models - splitovermpi T) in the log file you will also be given the ratio of the source luminosity to the sum of the energy per photon packet times its weight near the start of every iteration. If everything is working correctly, this should be close to 1.

For further information or if it not working properly, please contact Tom Haworth (haworth@astro.ex.ac.uk)

5.4.7 Periodic Boundary Conditions(domain decomposed models only)

In some cases you might want periodic boundary conditions. That is, photons that leave the grid at one side, re-enter the grid at the opposite side. An example of a case where this would be useful is to provide better signal to noise for cells near the edge that are otherwise not symmetrically sampled. To avoid infinite looping of low energy photons, packets are only allowed to cross a periodic boundary once. To include periodic boundary conditions, which are turned off by default, simply include the following in your parameters file:

```
periodicX T
periodicY T
periodicZ T
```

for the x, y and z boundaries respectively. Note that this only works for domain decomposed photoionization models (splitovermpi = T).

5.4.8 Customizing Atomic Species

You can specify what species you want present in a model, as well as their relative abundances. At present you have to include Hydrogen, if this is all that you want then include:

```
hOnly T
usemetals F
```

in the parameters file. To include just Hydrogen and Helium, you need the following

```
usemetals F
```

(hOnly is defaulted as False). Finally, to include metals as well simply to nothing as they are included by default.

Default abundances of metals are set to those used in the Lexington benchmark (relative to Hydrogen):

Species	**Default abundance (relative to Hydrogen)**
H	1.
He	0.1
C	22.e-5
N	4.e-5
O	33.e-5
Ne	5.e-5
S	0.9e-5

To customize these abundances, modify the above figures using the following names in your parameters file:

```
h_abund
he_abund
c_abund
n_abund
o_abund
ne_abund
s_abund
```

5.4.9 Retrieving data from .grid files

.grid files from domain decomposed photoionization/radiation hydrodynamics models can be used to dump vtk/vtu files by including

```
justdump T
```

in the parameters file. When executed the grid will be read in and immediately dump a vtk/vtu file for that grid, upon completion of the dump torus will abort.

Note: doing domain decomposed photoionisation calculations will not not work if torus has been compiled with the openmp=yes flag

Complete:
Responsible:
Author: Thomas Haworth 05 Apr 2011
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 05 Apr 2011
Not yet reviewed

5.4.10 Creating Synthetic Images

See the dust continuum model page for how to set up generic information about images (e.g. size, inclination).

For images generated by the photoionisation routines there is an additional `imagetype` option, which specifies the type of image to be generated.

imagetype options are:

- freefree
- forbidden
- recombination
- dustonly

Complete:
Responsible:
Author: Thomas Haworth 10 Jun 2011
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 10 Jun 2011
Not yet reviewed

5.5 Hydrodynamics

TORUS performs hydrodynamics calculations using a Eulerian grid-based algorithm. It is flux conserving, uses Rhie-Chow interpolation and is total variation diminishing (TVD).

How To Include Hydrodynamics

To perform a Hydrodynamics calculation, simply include the line

```
hydrodynamics T
```

in your parameters.dat file. Because it only works using multiple threads in a domain decomposition as outlined in Running photoionization models you also need to include the following:

```
splitovermpi T
```

5.5.1 Flux limiting

A flux limiter prevents the formation of oscillations near shocks, whilst attempting to retain physical oscillations, in piecewise linear schemes. Use of a flux limiter is turned on by default and there are a number available for use:

Limiter type	Notes
superbee	This is the default flux limiter
minmod	
MC	
fromm	
vanleer	
donorcell	This is equivalent to turning the flux limiter off

To select a specific flux limiter, include the following in your parameters file:

```
limitertype minmod
```

and replace minmod with whichever of the limiters you want in the table above. Generally there should never be any need to use anything other than the superbee, however the others (particularly donorcell) can be useful when debugging and developing.

5.5.2 Rhie Chow Interpolation

In a scheme where pressure is calculated by considering the difference between the i+1th and i-1th cells we can be left with two decoupled pressure fields. This gives rise to a "checkerboard effect" and essentially leaves one with half the resolution that they are otherwise computing. To get around this, as TORUS had to, one can use Rhie Chow interpolation. This uses the pressure difference at a cell's own interfaces, calculated by averaging with its neighbours and thus removing any possibility of decoupling. Rhie-Chow interpolation is turned on by default, but can be turned off by using the following:

```
rhiechow F
```

5.5.3 Boundary Conditions

Hydrodynamics models require boundary conditions. These must be specified in order for your model to run. The following should be specified as appropriate in your parameters file:

```
xplusboundstring
xminusboundstring
yplusboundstring
yminusboundstring
zplusboundstring
zminusboundstring
```

If you are running a 1D model then you only need to specify the x boundaries, 2D models require specification of the x and z boundaries and 3D models require all of them. These boundary specifiers should be matched by your desired option for the boundary condition from one of the following strings:

Boundary string	Notes
reflecting	impinging material will be reflected from the boundary
periodic	material leaving through one boundary (e.g. +x) will re-appear at the opposite boundary (e.g. -x)
shock	
freeOutNoIn	material is allowed to flow freely off the grid, but none is allowed to flow onto the grid
constDenNoVel	a constant density, zero velocity boundary. AKA a brick wall.
inflow	material streams onto the grid from the inflow boundary
inflowGrad	Apply a gradient to the material streaming onto the grid

an example use is as follows:

```
xplusboundstring periodic
```

In order to use the inflow boundary condition with a gradient (inflowGrad), you will need to specify along which axis there is a gradient (e.g. in 3D is the gradient along the x boundary in the y or z directions?). Do this by setting one of

```
xslope
yslope
zslope
```

to T or F in your parameters file. At present the gradients are geometry specific and you cannot customise them in your parameters file. This will be updated before too long.

5.5.4 Artificial Viscosity

By default, artificial viscosity is turned on. You can turn it off by including:

```
useviscosity F
```

in your parameters file. And to provide a custom numerical value for the artificial viscosity (default is 3.d0) use

```
etaviscosity 9.d0
```

Where 9.d0 is whatever value you wish to use.

5.5.5 Saving time and space in large calculations

Torus dumps vtu and grid files periodically throughout hydrodynamics (and RHD) calculations. The vtu files are used for visualisation in VisIt and the grid files are used as checkpoints from which the calculation can be restarted. In long, high resolution calculations the storage demands and time spent writing all of these files can be a problem. In particular the grid files can get very large.

It is often the case that visualisation files are required more frequently than restart checkpoints. By using the

```
vtuToGrid
```

parameter you can specify the vtu to grid file ratio.

If you had a parameters.dat file which dumps both grid and vtu files every time t = 10, you could change

```
tdump 10.0
```

to

```
tdump 1.0
vtuToGrid 10
```

then you will get the same number of .grid files as in the original but 10 times more vtu files. If you had

```
tdump 10.0
vtuToGrid 10
```

then you would get the same number of vtu files as in the original parameters file but 10 time fewer .grid files.

Complete:
Responsible:
Author: Thomas Haworth 13 Dec 2010
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 13 Dec 2010
Not yet reviewed

5.6 Radiation Hydrodynamics

Radiation hydrodynamics models combine the domain decomposed photoionization and hydrodynamics routines in torus to perform simulations where the evolution of the radiation field and material dynamics are not independent.

The algorithm that torus uses performs an initial radiative transfer calculation to give the starting state of the grid. It then performs sequential hydrodynamics and radiative transfer steps.

To run a radiation hydrodynamics model, the following needs to be included in your parameters file:

```
radiationhydro T
hydrodynamics T
splitovermpi T
photoionphysics T
```

Radiation hydrodynamics models are domain decomposed and thus require the same number of processors as specified in Running photoionization models. It is also possible to scale up the number of processors used by running a hybrid MPI/openMP model. This requires one node for each of the processors that would normally be required. For example, a 3D job using 9 processors will now require 9 nodes and a 3D model using 65 processors will now require 65 nodes(!). In hybrid mode, the initial processors use one thread on the node and the rest contribute to the openMP part of the calculation. To run in hybrid mode just compile with `openMP=yes`, you will also need to modify your jobfile - contact your system administrator for help with this.

Complete: ☐
Responsible:
Author: Thomas Haworth 09 Jun 2011
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 09 Jun 2011
Not yet reviewed

5.7 TORUS-3DPDR

Torus is coupled with 3D-PDR (Bisbas et al. 2012, MNRAS, 427, 2100) code to model the chemistry of photodissociation regions (pdr's). This work has been done by Thomas Haworth (Cambridge/Exeter) and Thomas Bisbas (UCL). The details and testing of the coupled code are given in Bisbas et al. (2015), MNRAS, 454, 2828. The svn branch for torus_3dpdr is in .../branches/torus_3dpdr. Note that the svn version is not yet fully up to date, but this is being worked on...

TORUS-3DPDR highlights/summary

- Calculate the UV field everywhere on the grid using domain decomposed Monte Carlo photoionisation (photoionAMR_mod.F90). Gives temperature in ionised gas
- UV field is fed into the PDR modules
- PDR modules use the UV field to work out abundances and level populations of various species, as well as calculate the gas and dust temperature.
- Result is more accurate thermal properties (i.e. not going from fully ionised, $10^{4K,}$ to molecular 10K across only ~1 cell) as well as the means to produce powerful synthetic observations (the normal torus molecular calculations assume an abundance - I've not yet done a full comparison to figure out in what regime torus-3dpdr becomes essential).
- Can combine PDR with normal torus radiation hydrodynamics (this is done in a very simple 1D case in Haworth et al. 2015, but Haworth intends to pursue this area of research in the near future.)

Main parameters for the input deck are

```
pdr T
hlevel <integer>
UV_low <double>
UV_high <double>
```

where hlevel determines the number of healpix rays (12*4**hlevel rays) and UV_low/UV_high specify the lower and upper frequencies that are considered "UV" by the PDR calculation.

If you want to prescribe a UV field rather than calculate it directly you need to specify the %uv_vector in the geometry specification in amr_mod. For example, to set up a 10 draine UV field impinging from the -x direction you would specify

```
thisOctal%uvvector(subcell)%x = 10.*Draine/1.d10
     thisOctal%uvvector(subcell)%y = 0.d0
     thisOctal%uvvector(subcell)%z = 0.d0
```

The latter involves running a photoionization calculation that tracks the vectors of the UV field. I.e. use all the usual photoionization parameters plus

```
UV_vector T
```

You can then either read in the result and do the pdr stuff in a new calculation or call the pdr routines as part of the same single calculation (pdr T, photoioneq T...).

I'll add comments on doing radiation hydro with the PDR, as well as some tests to /torus/benchmarks shortly...

Complete: ☐
Responsible:
Author: Thomas Haworth 25 Oct 2013
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 25 Oct 2013
Not yet reviewed

Chapter 6

Defining radiation sources

6.0.1 Defining radiation sources

The torus grid may be illuminated by one or more radiation sources (normally stars). These sources provide what is essentially a boundary condition to the radiation field. Sources are described by a position, radius, effective temperature, mass, and a spectral energy distribution.

The number of radiation sources in the defined by the `nsource` integer keyword which must be specified in the input deck. Each source is then defined by a set of keywords appended with the source number (i.e. `radius1` refers to the radius of source number 1, `radius2` is the radius of the second source etc).

The source position needs to be specified. The `sourcepos` keyword expects three double precision numbers corresponding to the source's (x,y,z) coordinate (remember the code distance units are 10^{10}cm). If the source is placed on the edge or corner of a grid then all photon packets emerge in the grid space and are appropriately weighted. The source mass is specified by `mass` and the units are solar masses. Similar the source radii are specified using the `radius` keyword, with units of the solar radius. The effective temperature of the source is given in Kelvins with `teff`.

The SED of the source can be defined in a number of ways by using the `contflux` keyword. Setting `contflux blackbody` gives a Planck function SED. Setting `contflux somefilename.dat` will read the SED from `somefilename.dat`. The SED file should contain two columns, the first of which should be wavelength or frequency and the second flux. Wavelengths should be in angstroms and fluxes should be in either erg/s/cm^2/angstrom (for wavelength space) or erg/s/cm^2/Hz (for frequency space files). The code works out itself whether the file is wavelength or frequency space and will warn you if it has to perform a conversion. The flux file should be space delimited and should not contain tabs.

Sources can also be read in from a file in the TORUS directory. This file is called `inputstar.dat` and contains the following information in a table: Stellar Mass (in Msol), Stellar Temperature (Kelvin), Stellar Radius (in Rsol) and Position in x, y and z (in 10^{10}cm). To use this file one must use the parameter `readstars` and set `nstar` to the number of stars.

The final option for the `contflux` keyword is `contflux kurucz` which will interpolate in the Kurucz LTE model atmosphere grid using the given `teff` and the mass and radius to get *log g*. You can now also specify `contflux tlusty` to use the non-LTE metal line blanketed models from the tlusty code (the full SED). The default metallicity for the tlusty models used by torus is solar, however you can use half or double solar by specifying `metallicity 0.5` or `metallicity 2.0` respectively.

In some modes each source will be defined by a surface across which the temperature (and therefore surface brightness) can vary. This is useful to simulate accretion hot spots for example.

Parameter	**Units**	**Description**
nsource		The number of radiation sources
sourcepos	10 10cm	(x,y,z) coordinate of the source
mass	solar masses	The source mass
radius	solar radii	The source radius
teff	K	The source effective temeprature
contflux		SED distribution, either `blackbody`, `kurucz`, `tlusty` or a filename
metallicity	solar metallicity	default is 1.0, other options are 0.5 or 2.0. Only works with tlusty grids.
readstars		Read in stars from `inputstar.dat`
nstar		Number of stars to read in

Complete:
Responsible:
Author: Tim Harries 06 Jan 2011
Contributors: Josh Hamilton, Tim Harries
Last significantly modified by: Josh Hamilton 25 Nov 2014
Not yet reviewed

Chapter 7

Adaptive Mesh Refinement

7.0.1 Customizing the AMR grid

Any problems, contact Tom Haworth (haworth@astro.ex.ac.uk).

The computational grid can automatically adapt to provide improved resolution for lower computational cost than refining the entire grid. This can be applied to both models that use hydrodynamics and those that perform photoionization calculations iteratively.

To use the adaptive mesh simply have mindepthamr and maxdepthamr set to different values in the parameters file and it will be used by default. If you want to only refine, unrefine, or neither (i.e. a fixed staggered grid) then simply use one or both of the following:

```
dorefine F
dounrefine F
```

these are set to T by default.

Grid initiation

The starting grid cell configuration for a given model is hardwired into the code. You can also modify the initial grid to capture the initial density configuration by adding:

```
gridshuffle T
```

to your parameters file.

BUG WARNING: At present this only works when the difference between your minDepthAMR and maxDepthAMR is EVEN. For example, you can use min 4, max 6 or min 4, max 8, or min 5, max 7 however min 4 max 7 would fail. I don't know why this is yet.

Options for hydrodynamic's models

There a number of options for refining the AMR grid in models that incorporate hydrodynamics. By default the grid refines and unrefines based on the density gradient. The gradient limit before refinement can be set using

```
amrtolerance <dble>
```

To refine cells that contain mass exceeding the limit of 10**-5 solar mass, include:

```
refineonmass T
```

in the parameters file. The limit for this mass can be specified using

```
masstol <dble>
```

It is also possible to refine the grid by considering the temperature, rhoe, speed and ionization fraction gradients via:

```
refineontemperature T

refineonrhoe T

refineonspeed T

refineonionization T
```

The allowed gradient limit can be specified using the following:

speed

```
amrspeedtol <dble>
```

temperature

```
amrtemperaturetol <dble>
```

energy

```
amrrhoetol <dble>
```

and ionization fraction

```
amrionfractol <dble>
```

for each of which the default value is 1.d-3.

Shock Capturing

Shocks are identified by looking at the ration in pressure gradients over two cells to the pressure gradient over 4 cells. If this ratio is greater than 1/3 then refinement will occur. This uses a modified version of the flattening identifier from Fryxell et al (2000, ApJ , 131, 273) .

To include this in your models, add:

captureshocks T

to the parameters file. At present this is turned off by default.

Coarse To Fine Flux Interpolation

When advecting from a coarse cell into two finer ones, the ambient flux gradient can be used to modify the fine cell incoming fluxes and capture the flow more fully. This is enabled by default, however it can be turned off by specifying:

```
fluxinterp F
```

Options for radiative transfer models

At present the grid will only refine adaptively between iterations for domain decomposed photoionization models. To do this, include

```
dophotorefine T
```

In the parameters file.

At the end of each iteration, the gradient in ionization fraction will be used to assess whether or not the grid should be refined. Be sure to include

```
refineonionization T
```

too or nothing will happen. Again, you can modify amrionfractol in the parameters file to change the ionization fraction gradient over which the AMR grid will refine.

Complete:
Responsible:
Author: Thomas Haworth 15 Aug 2011
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 15 Aug 2011
Not yet reviewed

Chapter 8

Geometries

8.0.1 Protoplanetary disc geometry

Use the `shakara` geometry to set up a flared protostellar disc. The parameters of the disc can be specified using the keywords listed in the table below. If a gap (low density annulus) is specified then the mass of the disc (integrated over the disc volume) will not match `mdisc`.

Parameter	**Units**	**Description**
mdisc	solar masses	The total mass of the disc (gas plus dust)
alphadisc		Disc density goes as r -alpha
betadisc		Disc scaleheight goes as r beta
rinner	stellar radius (source1)	Inner disc radius
router	AU	Outer disc radius
height	AU	Disc scaleheight at 100 AU
rgapinner	AU	Inner radius of gap
rgapouter	AU	Outer radius of gap
rhogap	g/cm 3	Minimum density in the gap
smoothinneredge		Exponential decay of density at the inner edge

Complete:
Responsible:
Author: Alicia Aarnio 07 Jan 2011
Contributors: Alicia Aarnio
Last significantly modified by: Alicia Aarnio 07 Jan 2011
Not yet reviewed

8.0.2 ttauri geometry

Setting `geometry ttauri` provides a geometry for protostellar accretion and outflow. This geometry can be broken down into three components:

- The magnetosphere. A dipole magnetosphere that can have a dipole offset compared to the rotation axis (always the z-direction). The footprints of the magnetospheric accretion stream produce hotspots on the surface of the protostar (assumed to be source number one, placed at the origin).
- The disc wind. A bipolar-style outflow whose direction and launch annulus can be altered (per Knigge et al., 1995).
- A dusty accretion disc. This is a standard alpha disc with a curved (Isella and Natta style) inner rim.

Each component can be present/absent from the geometry. Radiative equilibrium can be solved for the dusty disc, while statistical equilibrium for pure hydrogen can be solved for the wind and magnetosphere. Subsequently data cubes for particular line transitions can be calculated, along with emission line spectra. The continuum emission from the inner rim can be included as a boundary condition for the radiation field.

8.0.3 Magnetosphere parameters

Parameter	Units	Description
ttaurimag		Set to T to include a T Tauri magnetosphere
mdotpar1	solar masses per year	The accretion rate
ttaurirstar	solar radii	The radius of the protostar (must be same as radius1)
ttaurimstar	solar masses	The mass of the protostar (must be the same as mass1)
ttauririnner	stellar radii	The inner radius of the magnetosphere at the midplane
ttaurirouter	stellar radii	The outer radius of the magnetosphere at the midplane
ttauridiskheight	stellar radii	??
mdottype	constant	constant accretion rate
isotherm		Set to T to use an isothermal temperature structure for the magnetosphere
isothermtemp	Kelvin	Isothermal temperature of the magnetosphere
usehartmanntemp		Set to T to use a Hartmann-style temperature structure for the magnetosphere
maxharttemp	Kelvin	Maximum of Hartmann temperature
dipoleoffset	degrees	Dipole offset
thotspot	Kelvin	Hot spot temperature
lxoverlbol	Bolometric Luminosities	X-ray luminosity
maxcellmass	grams	Maximum cell mass after splitting
mdottype		Default is constant
enhance		Set to T to include accretion enhancement
mdotparN	solar masses per year	N ranges from 1-6; for constant accretion, only set mdotpar1
holeradius	stellar radii	Size of hole in geometrically thin, optically thick disk
curtainsphi1s	degrees	First accretion curtain, start angle phi
curtainsphi1e	degrees	First accretion curtain, end angle phi
curtainsphi2s	degrees	Second accretion curtain, start angle phi
curtainsphi2e	degrees	Second accretion curtain, end angle phi

8.0.4 Disc wind parameters

Parameter	Units	Description
ttauriwind		Set to T to add a disc wind
DW_theta	degrees	Launch angle for wind (90 is perpendicular to disc)
DW_Twind	Kelvin	Isothermal temperature of the wind
DW_Tmax	K	Temperature of disc at inner radius
DW_Mdot	solar masses per year	Wind mass-loss rate
DW_alpha	-	Exponent in the mass-loss rate per unit area
DW_beta	-	Exponent in beta-velocity law
DW_gamma	-	Exponent in the disc wind temperature law
DW_f	-	Scaling of asymptotic terminal velocity; default 2.0
DW_Rs	DW_Rmin	Effective wind acceleration length; default 50xRmin
oldwindunits		If T, use old units; default: F. Old/new listed below as affected
DW_Rmin	ttaurirouter/stellar radii	Minimum radius for launch annulus
DW_Rmax	ttaurirouter/stellar radii	Maximum radius for launch annulus
DW_d	DW_Rmin/stellar radii	Wind source displacement radius

8.0.5 Dusty disc parameters

Parameter	**Units**	**Description**
ttauridisc		Set to T to add a dusty alpha disc
rinner	stellar radii	The inner disc radius
router	AU	The outer disc radius
rsub	stellar radii	Sublimation radius
disctemp	K	Disc temperature inside sublimation radius
discpower		Disc temperature power law index inside sub radius
height	AU	The disc scale height at 100 AU
mdisc	solar masses	The disc mass
alphadisc		The density goes as rho -alpha
betadisc		The scaleheight goes as h beta
epsilon		ratio of Planck mean opacity at stellar and inner disc temperatures
ttauriwarp		Set to T to include warped disc around magnetosphere
hoverr		Warped disc H/R
midplanetemp	K	Blackbody Disc temperature inside sub radius
midplanepower		Blackbody disc power law index inside sub radius

Complete:
Responsible:
Author: Tim Harries 11 Jan 2011
Contributors: Tim Harries
Last significantly modified by: Tim Harries 11 Jan 2011
Not yet reviewed

Chapter 9

Using Torus with SPH data

9.1 Generating Torus grids from SPH data

The Torus AMR grid can be set up from density, temperature and velocity fields derived from SPH particles. This can either be from an SPH dump file, which is read in by Torus or can be passed through the subroutine interface in TorusMod. Several different SPH dump file formats can be read and the file format needs to be specified with the `inputFileFormat` parameter. The formats currently handled are binary files from the SPH-NG code (binary), binary files from Gadget2 (gadget2) or ascii files generated by `splash to ascii` (ascii). Gadget2 dumps can be either the default format or block labelled format (Torus will work out which of these formats the dump is in). If you are using an ascii file from splash then this should be in code units rather than physical units (this is set with option d7 in splash). The geometry `sphfile` will set up a grid without velocities stored at octal corners, whereas the geometry `theGalaxy` will set up a grid with velocities stored at octal corners (e.g. for line calculations).

The grid is constructed in two stages. Firstly the grid is split to achieve the required resolution, then the cells are populated with values calculated from the SPH particles.

There are three main conditions used to split the grid. Firstly the grid can be split if the mass within a cell exceeds a specified threshold (the threshold is set by the parameter `limitscalar`, don't use the default value for this parameter). Secondly a cell can be split if the density contrast within the cell exceeds a given threshold (specified by `limittwo`). Thirdly it is possible to take into account the velocity difference between particles in a cell and split that cell if a threshold is exceeded (controlled by the parameter `doVelocitySplit`). The velocity splitting condition was set up specifically for the Rundle et al calculations and will most likely require some modification if it is used for anything else. If you require a high level of refinement, which captures all the detail of the original SPH simulation, then set `sphonepercell T` which will split any grid cell which contains more than one SPH particle.

Cells on the AMR grid are populated with values using an SPH interpolation sum. Using the `kerneltype` parameter the interpolation kernel can be selected to be either exponential (Gaussian) or spline. However note that Monaghan (1992) states that a Gaussian kernel is best when a physical interpretation of the SPH equations is required (as in this case). It is possible to control when interpolated values are normalised by the sum of the kernel weights (either with `sphNormLimit` or `useHull`). Normalisation is required to reduce noise at points between SPH particles but should not be done at points significantly outside the particle distribution. If you find gaps in your density distribution, or a large discrepancy between the mass on the grid and the mass of the SPH particles, then you may need to adjust this parameter. To speed up the calculation of the interpolated values not all the SPH particles are used. The parameters `hcritPercentile` and `hmaxPercentile` control the selection of which particles are included in the interpolation sum, but the default values should work sensibly.

Sink and star particles present in the SPH dump can be read in and stored, with the possibility of using them as sources in a subsequent calculation. The logical parameter `discardsinks` tells Torus to not store sink/star particles. Unless you specifically want to keep sinks/stars set this to false (which is the default).

If the SPH simulation included chemistry then you can set `convertrhotohi T` to convert the total density to HI density and store this on the AMR grid (e.g. if you are generating a 21cm data cube from a run with chemistry). If the SPH run included chemistry, and you want to store CO and H2 information on the Torus grid, then set `sphwithchem T`. If you are reading in H2 or CO data from an ASCII dump file it is possible to specify which column to use for the H2/CO fraction with the parameters `ih2frac` and `iCOfrac`

Parameter	Default value	Description
sphdatafilename	None	Name of file containing SPH data
inputFileFormat	ascii	Format of SPH file (ascii, binary or gadget2)
limitscalar	1000.0	Maximum mass per cell (g)
limittwo	0.0	Density contrast splitting condition
doVelocitySplit	False	Split grid on velocity contrast
sphonepercell	False	Split if there is more than one particle in an AMR cell
sphNormLimit	0.5	Normalise by sum of weights above this threshold
kerneltype	0	0 is exponential,1 is spline
useHull	false	Use hull particle method
hcritPercentile	0.80	Percentile for hcrit
hmaxPercentile	0.99	Percentile for hmax
discardsinks	false	Sink particles are not stored
convertrhotohi	false	Convert total density to HI density
sphwithchem	false	Store CO and H2 information
ih2frac	11	Column of ASCII file which contains H2 fraction
iCOfrac	15	Column of ASCII file which contains CO fraction

Complete:
Responsible:
Author: David Acreman 13 Feb 2009
Contributors: David Acreman
Last significantly modified by: David Acreman 13 Feb 2009
Not yet reviewed

9.2 SPH-Torus

SPH-Torus is a combination of Torus with Matthew Bate's SPH-NG code (see the benchmarking and methods paper which applied the code to modelling a circumstellar disc.) To use SPH-Torus you need to build Torus as a library; to do this run `make lib` instead of the usual `make` command. When the build has completed you will have a library called `libtorus.a` which can be statically linked to the SPH code. The script `build_sphtorus.sh` automates the build of both Torus and SPH-NG.

Complete:
Responsible:
Author: David Acreman 13 Feb 2009
Contributors: David Acreman
Last significantly modified by: David Acreman 13 Feb 2009
Not yet reviewed

Chapter 10

Galactic Plane Surveys

10.1 Galactic Plane Surveys

10.1.1 Overview

Torus can be used to generate synthetic Galactic Plane Surveys (GPS), which produces data cubes in l-b-v co-ordinates (Galactic longitude, Galactic latitude, velocity). The process is based on the ray tracing method used to generate synthetic molecular line observations but in the GPS case the observer is placed inside the model grid and a series of diverging ray traces are performed (rather than the parallel ray traces used in the far-field case). The ray tracing method is not subject to Monte-Carlo noise but at the cost of not representing scattering processes. At the time of writing this module is used with SPH data as input and produces either 21cm or CO data cubes as output. When using SPH data the geometry should be `theGalaxy` to ensure that velocities at octal corners are set up.

To generate a GPS data cube set `datacube T` and `internalView T` in the parameters file. For a 21cm cube set `h21cm T` or for a CO cube use `molecularphysics T`. If you want to generate a CO cube assuming LTE then set `setupMolecularLteOnly T` in the parameters file.

Next you need to describe the size and axis ranges of the data cube. These parameters are summarised in the following table

Parameter	**Default value**	**Description**
centrevecx	None	Galactic longitude of the centre of the cube
centrevecy	None	Galactic latitude of the centre of the cube
npixels	None	Number of pixels in l and b dimensions
imageside	None	Size of cube in l and b dimensions (degrees)
cubeaspectratio	1.0	Aspect ratio (l/b)
maxVel	None	Maximum velocity channel (km/s)
minVel	-maxVel	Minimum velocity channel (km/s)
nv	None	Number of velocity channels

A number of parameters are required to describe where the observer is located (i.e. location within the Torus grid), and what velocity the observer is given. If `obsVelFromGrid` is true then Torus will set the observer's velocity to the value found on the AMR grid at the observer's location. This value can be modified by specifying one or more of `intDeltaVx`, `intDeltaVy` and `intDeltaVz` which are added to the observer's velocity. If you want to specify the observer's velocity yourself then set `obsVelFromGrid F` and give the required values in `intDeltaVx`, `intDeltaVy` and `intDeltaVz`.

Parameter	**Default value**	**Description**
intPosX	0.0	x-postion of observer
intPosY	2.2e12	y-position of observer
intPosZ	0.0	z-position of observer
obsVelFromGrid	None	Set the observer velocity from AMR grid.
intDeltaVx	0.0	Additional x-velocity (km/s)
intDeltaVy	0.0	Additional y-velocity (km/s)
intDeltaVz	0.0	Additional x-velocity (km/s)

There are some parameters which describe how the ray trace is carried out. Setting `densitysubsample T` makes Torus use a linear interpolation to calculate the density at a given location within a grid cell. The result is a smoother looking data cube but at the expense of increased run time. If the grid resolution is very high, or the spatial resolution of the data cube is low, the ray traces may undersample the variability of the grid. In this case you can set nSubpixels to a number greater than 1, to make Torus do multiple ray traces along slightly different directions, when calculating a pixel in the data cube. If you want to increase the line width by adding a specified "turbulent" velocity then this can be set using the vturb parameter. If this is not added then the line width will be the thermal line width.

Parameter	**Default value**	**Description**
densitysubsample	False	Use a linear interpolation to determine density
nSubpixels	1	Number of ray traces per pixel
vturb	0.0	Turbulent velocity (km/s)

Lastly there are some parameters to control the data cubes which are written out. `datacubefile` specifies a suffix (e.g. gal.fits) which makes up the last part of the data cube filename; a prefix is added to describe what is in the file. There are two parameters which enable additional output. `splitCubes` writes out the positive and negative components of the ray trace separately, which is relevant when looking for self-absorption, and `wantTau` writes out an optical depth cube.

Parameter	**Default value**	**Description**
datacubefile	None	File name suffix
splitCubes	False	Write out positive and negative components separately
wantTau	False	Write out optical depth cube

10.1.2 Column density calculations

Setting `nv 1` causes Torus to run in a mode where it calculates three different column densities, rather than data cubes. This is designed to work for runs with chemistry where there is information on molecular hydrogen fraction and CO abundance. Separate FITS files are written containing H2 column density (prefix nCol_H2_), CO column density (prefix nCol_CO_) and a third file (prefix nCol_) which contains the column density of either HI or total hydrogen (depending on the value of convertRhoToHI).

10.1.3 Runs with chemistry

If you are using an SPH run with chemistry information then some care is needed when setting the values of `convertRhoToHI` and `sphWithChem`. If you are generating a 21cm cube then use `convertRhoToHI T` and `sphWithChem F`; this uses HI density for calculating emissivity and opacity but does not store CO information. If you are generating CO cubes then use `convertRhoToHI F` and `sphWithChem T`; this stores CO data but doesn't convert total density to HI density (which is better for density sub-sampling). When calculating column density only (i.e. the `nv 1` option) then use `sphWithChem T` to store CO data and set `convertRhoToHI` according to what you require in the third column density file.

10.1.4 Particle file

In addition to the data cubes Torus will also write out a file containing additional information mapped on to the SPH particles. This file is called `particle_dI.dat` and contains information on which l-b-v point in the data cube the particle is associated with. There is also information on what effect this particle had on the radiative transfer calculation (the change in intensity due to the grid cell which contains the particle) to enable material associated with self-absorption to be identified. The `particle_dI.dat` file can be read with `asplash` and the `columns` file written by Torus will ensure that splash can correctly identify the columns in the file.

10.1.5 Recommended build options

Several large modules are not required for Galactic plane survey runs so it is recommended to build with `atomic=no hydro=no photoion=no xray=no`. This will reduce the memory use slightly, due to excluding statically allocated octal components, and reduce the time taken to build an executable.

Complete: ☐
Responsible:
Author: David Acreman 16 Sep 2013
Contributors: David Acreman
Last significantly modified by: David Acreman 16 Sep 2013
Not yet reviewed

Chapter 11

Torus input parameters

All of the non geometry specific parameters.

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

Torus parameters				
Parameter	**Default value**	**Default Units**	**Unit Type**	**Description**
outputfile	none		N/A	The output grid filename
inputfile	none		N/A	The input grid filename
readgrid	false		N/A	Read in an input grid (required)
writegrid	false		N/A	Write an output grid (required)
justdump	false		N/A	Dump a vtk/vtu file for a given .grid
amr1d	false		N/A	The AMR grid is 1-dimensional
amr2d	false		N/A	The AMR grid is 2-dimensional
amr3d	false		N/A	The AMR grid is 3-dimensional
dorefine	true		N/A	Allow the AMR grid to refine adaptively
dounrefine	true		N/A	Allow the AMR grid to coarsen adaptively
mindepthamr	5		N/A	Minimum refinement depth of AMR grid
maxdepthamr	31		N/A	Maximum refinement depth of AMR grid
amrgridsize	1000.	10 10cm	distance	The linear size of the AMR grid
amrgridcentrex	0.	10 10cm	distance	AMR grid centre x-coordinate
amrgridcentrey	0.	10 10cm	distance	AMR grid centre y-coordinate
amrgridcentrez	0.	10 10cm	distance	AMR grid centre z-coordinate
griddistancescale	1.d10	cm		Distance scale of grid (required)
cylindrical	false		N/A	Grid uses 3D cylindrical coordinates
geometry	sphere		N/A	Name of model to run
splitovermpi	false		N/A	Split grid across threads
binaryxml	true		N/A	Use binary XML VTK files
refineonmass	false		N/A	Refine AMR grid using cell mass
refineontemperature	false		N/A	Refine AMR grid using temperature gradient
refineonionization	false		N/A	Refine AMR grid using ionization fraction gradient
dophotorefine	false		N/A	Refine AMR grid between photoionization iterations
amrtolerance	1.d-3		N/A	Maximum gradient before refinement
amrtemperaturetol	1.d-3			Maximum temperature gradient before refinement
amrspeedtol	1.d-3			Maximum speed gradient before refinement
amrionfractol	1.d-3			Max. ionization fraction gradient before refinement
amrrhoetol	1.d-3			Maximum temperature gradient before refinement
masstol	1.d-5 solar masses	g	mass	Maximum allowed mass contained in one cell

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

timeunit	1.d0		N/A	Code unit of time
lengthunit	1.d0		N/A	Code unit of length
massunit	1.d0		N/A	Code unit of mass

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

nmonte	0		N/A	Defines number of photon packets between computational domains
hydrodynamics	false		N/A	Perform a hydrodynamics calculation
cfl	0.3		N/A	Courant-Friedrichs-Lewy constant
limitertype	superbee		N/A	Flux limiter to use
rhieChow	true		N/A	Use Rhie-Chow interpolation
fluxinterp	false		N/A	Do fine to coarse flux interpolation (incomplete)
useviscosity	true		N/A	Use artificial viscosity
xplusboundstring	null			Positive x direction boundary condition
xminusboundstring	null			Negative x direction boundary condition
yplusboundstring	null			Positive y direction boundary condition
yminusboundstring	null			Negative y direction boundary condition
zplusboundstring	null			Positive z direction boundary condition
zminusboundstring	null			Negative z direction boundary condition
tstart	0.d0	s	time	Start time in simulation
tend	1.d10	s	time	End time in simulation
tdump	0.d0	s	time	Time between data dumps in simulation

Source parameters

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

nsource	1			Number of sources (stars)
pointsource	false			Source object is pointlike
radiusj	1.d0	Solar radii		Radius of jth source
teffj	1.d0	K	temperature	Effective temperature of jth source
massj	1.d0	Solar masses		Mass of jth source
contfluxj	none			Continuum flux file for jth source
sourceposj	0.d0,0.d0,0.d0	10 10cm		Position of jth source
velocityj	0.d0,0.d0,0.d0	km/s		velocity of jth source
probsourcej	0.d0			Probability of photon packet from jth source
readstars	false			Read stars from file "inputstar.dat"
nstar	0.d0			Number of stars to read in

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

photoionphysics	false		N/A	Include photoionization physics in calculation
quickthermal	false		N/A	compute photoionization equilibrium
usemetals	true		N/A	Use metals (Z>2) in photoionization calculation
hOnly	false		N/A	Hydrogen only model
nodiffuse	false		N/A	Turn off the diffuse radiation field
monochromatic	false		N/A	Use a monochromatic (13.6+10 -5eV) radiation field
bufferCap	5000		N/A	Number of photon packet stacks accommodated in "to send" buffer
periodicX	false			Periodic photon boundaries in the x direction
periodicY	false			Periodic photon boundaries in the y direction
periodicZ	false			Periodic photon boundaries in the z direction
biasToLyman	false			Increase sampling of ionizing photons
biasMagnitude	100.d0			Magnitude of bias towards ionizing photons
binPhotons	false			Dump a spectrum formed from processed photon packets
h_abund	1.	X(H)/X(H)		Hydrogen abundance
he_abund	0.1	X(He)/X(H)		Helium abundance
c_abund	22.e-5	X(C)/X(H)		Carbon abundance
n_abund	4.e-5	X(N)/X(H)		Nitrogen abundance
o_abund	33.e-5	X(O)/X(H)		Oxygen abundance
ne_abund	5.e-5	X(Ne)/X(H)		Neon abundance
s_abund	0.9e-5	X(S)/X(H)		Sulphur abundance

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" header-color="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

radiationHydrodynamics	false			Perform a radiation hydrodynamics calculation
supernovae	false		N/A	Consider supernova feedback
stellarwinds	false		N/A	Consider stellar winds feedback

Output parameters

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" header-color="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

Image output				
image	false			Calculate an image
nphotons	10000			Number of photons in image
distance	100.	pc		Grid distance from observer
nimage	1			Number of images to calculate
imageaxisunits	AU			Axis units for image (AU,pc,cm,arcsec)
imagesize	whole grid	imageaxisunits		Size of image
imageaspect	1.0			Aspect ratio of image
polimage	false			Write polarization images
imagefile	none			Output image filename
lambdaimage	6562.8	Angstroms		Wavelength for monochromatic image
imagetype	none			Type of output image
npixels	200			Number of pixels per side in image
inclination	0.	degrees		Inclination angle
positionangle	0.	degrees		position angle
fitsbitpix	-32			BITPIX value for FITS images and cubes
Datacube output				
datacube	false			Calculate a data cube
ninc	1			Number of inclinations to calculate
inclinations	90.	degrees		Inclination angles
nlamline	1			Number of wavelengths to calculate
lamline	850.	Angstroms		Wavelengths to calculate
datacubefile				Name of output file
imageside	1.	10 10cm		Size of image on each side
npixels	50			Number of pixels per side in image
nv	50			Number of velocity bins
maxVel	1.d0	km/s		Range of velocities
positionangle	0.	degrees		Position angle
distance	100	parsec		Distance to source
SED output				
spectrum	false			Calculate a spectrum
nphotons	100900			Number of photons in SED*
inclinations	none			Inclination angles
posangs	0.0			Position angles
sed	false			Write spectrum as lambda vs lambda F_lambda
sised	false			Write spectrum as lambda (microns) vs lambda F_lambda (microns * W/m 2
jansky	false			Write spectrum in janskys

* Same parameter name (nphotons) used when generating an image but a different default value.

%TABLE sort="on" tableborder="0" cellpadding="1" cellspacing="3" headerbg="#D5CCB1" headercolor="#666666" databg="#FAF0D4, #F3DFA8" headerrows="2" %

dustphysics	false			Include dust physics in the calculation
dusttogas	0.01			Dust to gas ratio
ndusttype	1			Number of different dust types
grainTypeLabel	sil_dl			Dust grain type
grainFracLabel	1.			Grain fractional abundance
aminLabel	0.005	microns		Min grain size
amaxLabel	0.25	microns		Max grain size
qDistLabel	3.5			Grain power law
a0Label	1.0e20	microns		Scale length of grain size
pdistLabel	1.			Exponent for exponential cut off
iso_scatter	false			Isotropic scattering

Complete: ☐
Responsible:
Author: Ben Ambery Smith 06 Feb 2015
Contributors: Ben Ambery Smith, Josh Hamilton, Tim Harries, Thomas Haworth
Last significantly modified by: Tim Harries 04 May 2010
Not yet reviewed

11.1 Unit Directory

It is now possible to specify the units for certain variables in parameters.dat. This is still in the early stages, more quantities are being updated to include this all the time and the quantities for which units can be specified should be indicated in the A list of Torus input parameters section.

Any comments, problems or feedback email haworth@astro.ex.ac.uk .

Key Files:
- inputs_modV2.F90
- units_mod.f90

11.1.1 Available units

Case sensitive

Distances
- Default value: 10^{10} Centimetres
- cm - 10^{10} centimetres
- m - 10^{10} metres
- au - Astronomical units
- pc - parsecs
- rSol - solar radii

Wavelengths
- Default value: Angstroms
- A - Angstroms
- nm - Nanometers
- um - Microns
- mm - Millimetres

Dust grain sizes
- Default value: microns
- A - Angstroms
- nm - Nanometers
- um - Microns
- mm - Millimetres

Angles
- Default value: radians
- rad - radians
- deg - degrees
- arcmin - arcminutes
- arcsec - arcseconds

Masses

- Default value: grams
- g - grams
- mSol - solar masses
- kg - kilograms

Time

- Default: seconds
- s - seconds
- yr - years
- kyr - kiloyears
- Myr - megayears

Temperature

- Default: Kelvin
- K - Kelvin

Luminosity

- Default: Ergs per second
- ergsec - ergs per second
- lSol - solar luminosities

Velocities

- Default: cm s$^-1cms - cms^-1$

- ms - m s$^-1kms - kms^-1$
- c - fraction of light speed

Densities

11.1.2 Unit Types (for developers)

Torus performs calculations using cgs units, each kind of physical quantity has different units that torus will use. When adding new quantities to inputs_mod you are therefore required to specify a unit type, this is the third quantity passed in a call to getUnitDouble for example:

```
call getUnitDouble(keyword, sourceTeff(i), "temperature", cLine, fLine, nLines,
  &
           "Source temperature (K) : ","(a,f8.0,a)",1.d0, ok, .true.)
```

where the unit type is "temperature". Other unit types are:

- distance
- wavelength
- dust
- angle
- mass
- time
- temperature
- luminosity

More may be required.

Complete: ▭
Responsible:
Author: Thomas Haworth 18 Aug 2011
Contributors: Thomas Haworth
Last significantly modified by: Thomas Haworth 18 Aug 2011
Not yet reviewed

Chapter 12

Torus Publications

1 Boneberg, Dominika M.; Pani&#263;, Olja; Haworth, Thomas J.; Clarke, Cathie J.; Min, Michiel, 2016, Determining the midplane conditions of circumstellar discs using gas and dust modelling: a study of HD 163296, MNRAS, in press

2 Haworth, Thomas J.; Clarke, Cathie J.; Owen, James E., 2016, Rapid radiative clearing of protoplanetary discs, MNRAS, 457, 1905

3 Mackey, Jonathan; Haworth, Thomas J.; Gvaramadze, Vasilii V.; Mohamed, Shazrene; Langer, Norbert; Harries, Tim J., 2016, Detecting stellar-wind bubbles through infrared arcs in H II regions, A&A, 586, 114

4 Acreman, D. M.; Stevens, I. R; Harries, T. J., 2016, Modelling multiwavelength observational characteristics of bow shocks from runaway early-type stars, MNRAS, 456, 136

5 Bisbas, T. G.; Haworth, T. J.; Barlow, M. J.; Viti, S.; Harries, T. J.; Bell, T.; Yates, J. A., 2015, TORUS-3DPDR: a self-consistent code treating three-dimensional photoionization and photodissociation regions, MNRAS, 454, 2828

6 Haworth, T. J.; Harries, T. J.; Acreman, D. M.; Bisbas, T. G., 2015, On the relative importance of different microphysics on the D-type expansion of galactic H II regions, MNRAS, 453, 2277

7 Haworth, T. J.; Shima, K.; Tasker, E. J.; Fukui, Y.; Torii, K.; Dale, J. E.; Takahira, K.; Habe, A., 2015, Isolating signatures of major cloud-cloud collisions - II. The lifetimes of broad bridge features, MNRAS, 454, 1634

8 Haworth, T. J.; Tasker, E. J.; Fukui, Y.; Torii, K.; Dale, J. E.; Shima, K.; Takahira, K.; Habe, A.; Hasegawa, K., 2015, Isolating signatures of major cloud-cloud collisions using position-velocity diagrams, MNRAS, 450, 10

9 Pettitt, Alex R.; Dobbs, Clare L.; Acreman, David M.; Bate, Matthew R., 2015, The morphology of the Milky Way - II. Reconstructing CO maps from disc galaxies with live stellar distributions, MNRAS, 449, 3911

10 Bisbas, T. G.; Haworth, T. J.; Williams, R. J. R.; Mackey, J.; Tremblin, P.; Raga, A. C.; Arthur, S. J.; Baczynski, C.; Dale, J. E.; Frostholm, T.; Geen, S.; Haugblle, T.; Hubber, D.; Iliev, I. T.; Kuiper, R.; Rosdahl, J.; Sullivan, D.; Walch, S.; Wnsch, R., 2015, STARBENCH: the D-type expansion of an H II region, MNRAS, 453, 1324

11 Dai, Fei; Facchini, Stefano; Clarke, Cathie J.; Haworth, Thomas J., 2015, A tidal encounter caught in the act: modelling a star-disc fly-by in the young RW Aurigae system, MNRAS, 449, 1996

12 Harries, Tim J., 2015, Radiation-hydrodynamical simulations of massive star formation using Monte Carlo radiative transfer - I. Algorithms and numerical methods, MNRAS, 448, 3156

13 Duarte-Cabral, A.; Acreman, D. M.; Dobbs, C. L.; Mottram, J. C.; Gibson, S. J.; Brunt, C. M.; Douglas, K. A., 2015, Synthetic CO, H2 and H I surveys of the second galactic quadrant, and the properties of molecular gas, MNRAS, 447, 2144

14 Cleeves, L. Ilsedore; Bergin, Edwin A.; Alexander, Conel M. O.'D.; Du, Fujun; Graninger, Dawn; berg, Karin I.; Harries, Tim J., 2015, The ancient heritage of water ice in the solar system, Science, 345, 1590

15 Esau, Claire F.; Harries, Tim J.; Bouvier, Jerome, 2015, Line and continuum radiative transfer modelling of AA Tau, MNRAS, 443, 1022

16 Pettitt, Alex R.; Dobbs, Clare L.; Acreman, David M.; Price, Daniel J., 2014, The morphology of the Milky Way - I. Reconstructing CO maps from simulations in fixed potentials, MNRAS, 444, 919

17 Haworth, Thomas J.; Harries, Tim J.; Acreman, David M.; Rundle, David A, 2013, Assessing molecular line diagnostics of triggered star formation using synthetic observations, MNRAS, 431, 3470

18 Haworth, Thomas J.; Harries, Tim J.; Acreman, David M., 2012, Testing diagnostics of triggered star formation, MNRAS, 426, 203

19 Acreman, David M.; Dobbs, Clare L.; Brunt, Christopher M.; Douglas, Kevin A., The structure of HI in galactic disks: Simulations vs observations, MNRAS, 422, 241

20 Haworth, Thomas J.; Harries, Tim J., 2011, Radiation hydrodynamics of triggered star formation: the effect of the diffuse radiation field, MNRAS, 420, 562

21 Harries, Tim J., 2011, An algorithm for Monte Carlo time-dependent radiation transfer

22 Mayne, Nathan J.; Harries, Tim J., 2010, On the properties of discs around accreting brown dwarfs, MNRAS, 409, 1307

23 Rundle, David; Harries, Tim J.; Acreman, David M.; Bate, Matthew R. Three-dimensional molecular line transfer: a simulated star-forming region, MNRAS 407, 986

24 Acreman, David M.; Harries, Tim J.; Rundle, David A, Modelling circumstellar discs with three-dimensional radiation hydrodynamics, 2010, MNRAS, 403, 1143

25 Douglas, Kevin A.; Acreman, David M.; Dobbs, Clare L.; Brunt, Christopher M., A Synthetic 21-cm Galactic Plane Survey of a smoothed particle hydrodynamics Galaxy Simulation, 2010, MNRAS, 407, 405

26 Acreman, David M.; Douglas, Kevin A.; Dobbs, Clare L.; Brunt, Christopher M., Synthetic HI observations of a simulated spiral galaxy, 2010, MNRAS, 406, 1460

27 Pinte, C.; Harries, T. J.; Min, M.; Watson, A. M.; Dullemond, C. P.; Woitke, P.; Mnard, F.; Durn-Rojas, M. C., Benchmark problems for continuum radiative transfer. High optical depths, anisotropic scattering, and polarisation, 2009, A&A, 498, 967

28 Tannirkulam, A.; Monnier, J. D.; Harries, T. J.; Millan-Gabet, R.; Zhu, Z.; Pedretti, E.; Ireland, M.; Tuthill, P.; ten Brummelaar, T.; McAlister H.; Farrington, C.; Goldfinger, P. J.; Sturmann, J.; Sturmann, L.; Turner, N., A Tale of Two Herbig Ae Stars, MWC 275 and AB Aurigae: Comprehensive Models for Spectral Energy Distribution and Interferometry, 2008, ApJ 689, 513

29 Kurosawa, Ryuichi; Romanova, M. M.; Harries, Tim J., 3D simulations of rotationally-induced line variability from a classical T Tauri star with a misaligned magnetic dipole, 2008, MNRAS, 385, 1931

30 Tannirkulam, A.; Monnier, J. D.; Millan-Gabet, R.; Harries, T. J.; Pedretti, E.; ten Brummelaar, T. A.; McAlister H.; Turner, N.; Sturmann, J.; Sturmann, L., Strong Near-Infrared Emission Interior to the Dust Sublimation Radius of Young Stellar Objects MWC 275 and AB Aurigae, 2008, ApJ 677L, 51

31 Hatchell, J.; Fuller, G. A.; Richer, J. S.; Harries, T. J.; Ladd, E. F., Star formation in Perseus. II. SEDs, classification, and lifetimes, 2007, A&A, 468, 1009

32 Tannirkulam, A.; Harries, T. J.; Monnier, J. D., The Inner Rim of YSO Disks: Effects of Dust Grain Evolution, 2007, ApJ 661, 374

33 Monnier, J. D.; Tuthill, P. G.; Danchi, W. C.; Murphy, N.; Harries, T. J., The Keck Aperture-masking Experiment: Near-Infrared Sizes of Dusty Wolf-Rayet Stars, 2007, ApJ 655, 1033

34 Kurosawa, Ryuichi; Harries, Tim J.; Symington, Neil H., On the formation of H&#945; line emission around classical T Tauri stars, 2006, MNRAS, 370, 580

35 Kurosawa, Ryuichi; Harries, Tim J.; Symington, Neil H., Time-series Paschen-&#946; spectroscopy of SU Aurigae, 2005, MNRAS, 358, 671

36 Symington, Neil H.; Harries, Tim J.; Kurosawa, Ryuichi, Emission-line profile modelling of structured T Tauri magnetospheres, 2005, MNRAS, 358, 671

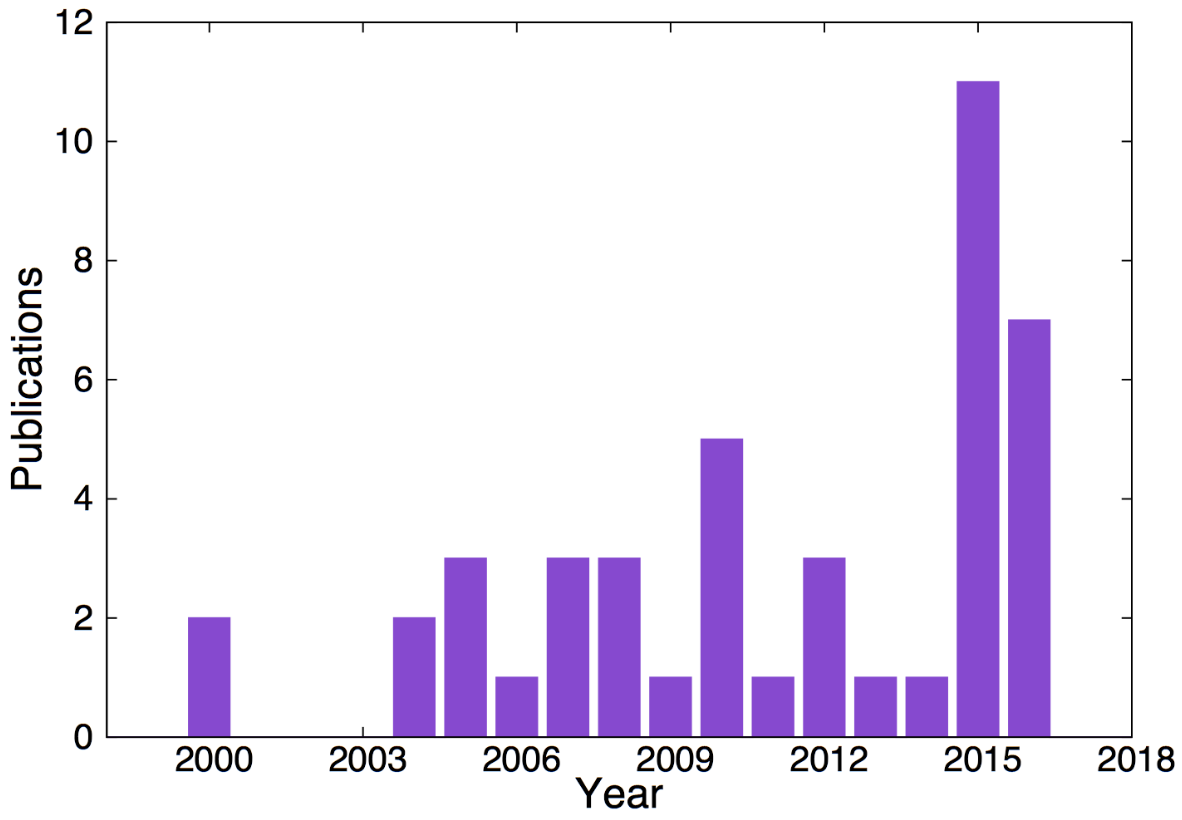
37 Vink, Jorick S.; Harries, T. J.; Drew, J. E., Polarimetric line profiles for scattering off rotating disks, 2005, A&A, 430, 213

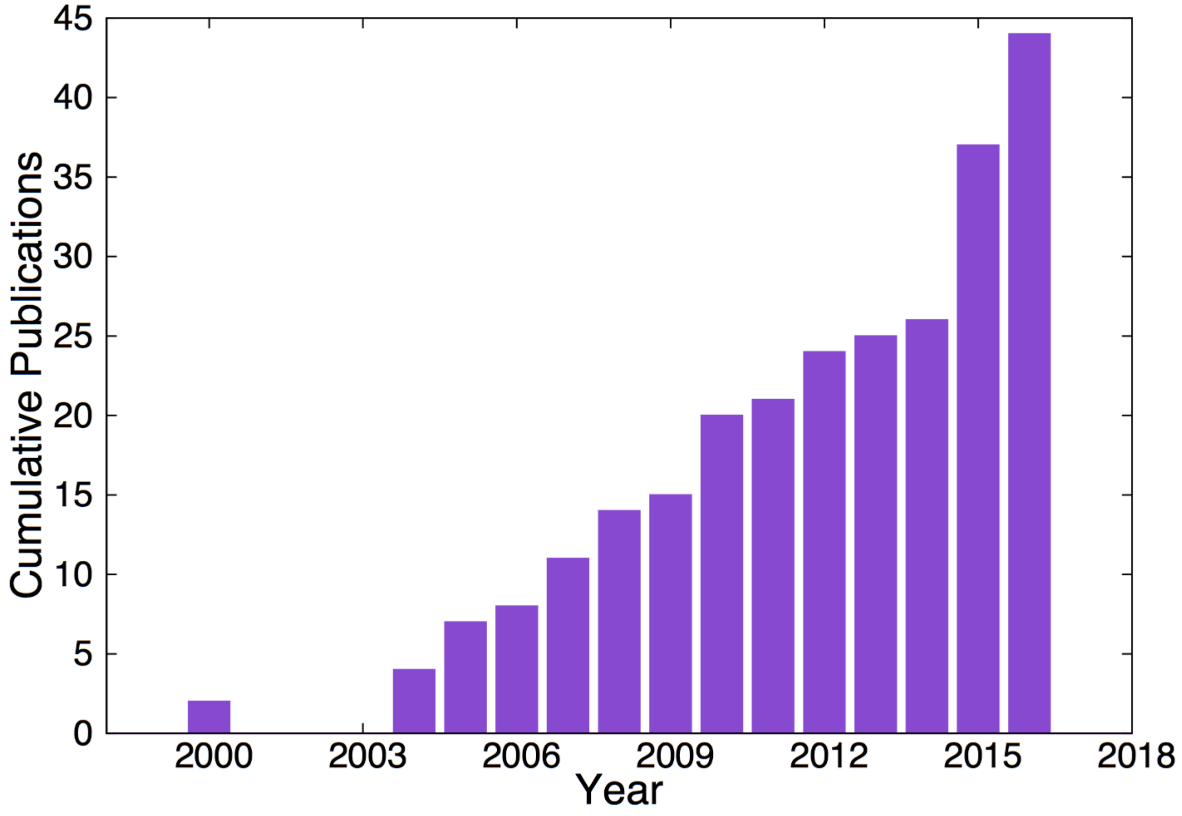
38 Kurosawa, Ryuichi; Harries, Tim J.; Bate, Matthew R.; Symington, Neil H., Synthetic infrared images and spectral energy distributions of a young low-mass stellar cluster, 2004, MNRAS, 351, 1134

39 Harries, Tim J.; Monnier, John D.; Symington, Neil H.; Kurosawa, Ryuichi, Three-dimensional dust radiative-transfer models: the Pinwheel Nebula of WR 104, 2004, MNRAS, 350, 565

40 Harries, T. J.; Babler, B. L.; Fox, G. K., The polarized spectrum of the dust producing Wolf-Rayet+O-star binary WR137, 2000, A&A, 361, 273

41 Harries, Tim J., Synthetic line profiles of rotationally distorted hot-star winds, 2000, MNRAS, 315, 722





Complete: ☐
Responsible:
Author: Tim Harries 05 May 2010
Contributors: Tim Harries
Last significantly modified by: Tim Harries 05 May 2010
Not yet reviewed