Sedona6 User Guide

Sep 16, 2018

Contents:

CHAPTER 1

Getting Started

1.1 Installing the Code

- Set the environment variable `SEDONA\_HOME` to point to the base directory of sedona. In bash, for example, you can add to your .bash profile the line:

```
export SEDONA_HOME=/Users/kasen/sedona6/
```

- Install lua. Source code can be found in the `src/external/` directory
- Download and install the Gnu Scientific Library (gsl) available at https://www.gnu.org/software/gsl/
- Download and install hdf5 from https://support.hdfgroup.org/downloads/index.html
- Go to the `src/` directory and edit the make.inc file so that CXX is your C++ compiler with options, and the variables point to the location of the installed libraries. Some make.inc files are already included for common systems (e.g., *make.inc.nersc*) and can simply be copied over to *make.inc*.
- Type **`make`**
- If compilation is successful, the executable file *sedona6.ex* will appear in the src/ directory. Copy this to the directory where you would like to run the code.

Python is currently used for plotting and testing scripts, but is not needed to build and run **sedona** itself (NB: Python may eventually replace lua for parameter files). Python 2.7 is being used. On linux to get the necessary packages try:

```
sudo apt-get install python-numpy python-scipy python-matplotlib python h5py
```

1.2 Running the Code

To get started with **sedona**, you can try running a simple transport calculation. Copy the executable file *sedona6.ex* into a directory where you want to run, for example type:

```
cp sedona6.ex examples/simple_examples/spherical_lightbulb/1D/
```

Change to that directory, and run the code by typing:

```
./sedona6.ex param.lua
```

where *param.lua* is the file name of the run time parameter file (if no file name is given, the name *param.lua* is assumed). See *Setting Runtime Parameters* for description of setting runtime parameters. At minimum, the parameter file must point to three files

- A model file (an ascii .mod file for 1D runs or a hdf5 file for multi-D) giving the physical conditions (e.g., density, composition) of the setup. See *Input Model Files* for description of formats.
- An atomic data file in hdf5 format. Such files are available in the `data` directory, with data compiled from various sources.
- A defaults file, giving the default settings for all runtime parameters. The standard is ‘‘ defaults/sedona_defaults.lau‘‘, although users can point to their own modified defaults file.

The code generates several files. The *plt_.?????.h5* files contain data in hdf5 format describing the grid properties (e.g., density, temperature, radiation field). If hdf5 tools are installed, type:

```
h5ls plt_00001.h5
```

to see the file contents. For 1D models, a *plt_?????.dat* gives some of this data in ascii format.

The code also generates a *spectrum_?.h5* hdf5 file (and for 1D models a *spectrum_?.dat* asci file) giving the output spectrum.

You will find in the `tools/` directory a python library file *sedona.py* that provides functions to read and analyze the data in the plot and spectrum files (NB: this needs to be developed)

CHAPTER 2

Basic Code Execution

Several example setups for different sorts of science runs are given in the `examples/` directory. Simpler tests are in the `tests/` directory.

sedona everywhere uses cgs units.

2.1 Input Model Files

Model file set the density, composition, temperature at each

2.2 Setting Runtime Parameters

Runtime parameters are currently set in the param file, by default is assumed to be called **param.lua**

The parameter files uses the Lua scripting language, which allows for using math expressions and function calls (may be replaced with a python option). One can also grab environment variables using, e.g.,:

```
sedona_home    = os.getenv('SEDONA_HOME')
```

which sets the local variable **sedona_home** based on the environment variable **$SEDONA_HOME**.

Default values of all parameters are set in a **defaults** file, which is also in the Lua language. The **param.lua** file must point to the defaults file using, e.g.,

```
defaults_file     = sedona_home.."/defaults/sedona_defaults.lua"
```

This points to the standard defaults file provided with **sedona**. (N.B. the .. notation means string concatination in Lua)

Scalar parameters are set as e.g.,:

```
tstep_time_stop  = 100.0
```

String parameters (such as filenames) are set using quotes, e.g.,:

```
transport_module = "monte_carlo"
```

Vectors parameters are currently set as e.g.,:

spectrum_nu_grid = {start,stop,delta}

where gives a uniform vector between values **start** and **stop** with spacing **delta**. To use a logarithmically spaced grid, we add an extra entry of 1:

```
spectrum_nu_grid   = {start,stop,delta,1}
```

where now the spacing between points is dx = x*delta

2.3 Controlling Time Evolution

Calculations in **sedona** can be run either as time evolving or steady-state models. This is controlled by time-stepping parameters

Table 1: Time Stepping Parameters

parameter	values	definition
tstep_max_steps	<integer>	Maximum number of time steps to take before exiting
tstep_time_start	<real>	Start time (in seconds)
tstep_time_stop	<real>	Stop time (in seconds)
tstep_max_dt	<real>	Maximum value of a time step (in seconds)
tstep_min_dt	<real>	Minimum value of a time step (in seconds)
tstep_max_delta	<real>	Maximum fractional size of a timestep – restricts dt to the specified value multiplied by the current time

Times are always in seconds.

2.4 Output Files

CHAPTER 3

Radiation Transport

3.1 Controlling Transport

Table 1: Radiation Transport Parameters

parameter	values	definition
transport_module	"monte_carlo"	What method to use for transport. Currently only monte carlo is implemented.
transport_nu_grid	<float vector>	Define frequency grid used for transport and opacities
transport_radiative_equilibrium	0 = no | 1 = yes	Whether to solve for radiative equilibrium
transport_steady_iterate	<integer>	Do a steady-state calculation with this number of iterations

Note: The last parameter should really be under tstep, since it controls time evolution not transport per se.

3.2 Controlling Output

Spectrum files and writeout files

3.3 Interaction Processes

3.3.1 Electron Scattering

3.3.2 Compton Scattering

3.3.3 Resonant Line Scattering

CHAPTER 4

Radiation Sources

4.1 Radioactivity

4.2 Thermal Emission

4.3 Radiating Core

Table 1: Radiating Core Parameters

parameter	values	definition
core_n_emit	<integer>	Number of particles to emit from core per time step (or iteration)
core_radius	<real>	Radius (in cm) of emitting spherical core
core_luminosity	<real> or <function>	Luminosity (in erg/s) emitted from core
core_temperature	<real>	Blackbody spectrum of core emission, if using blackbody emission
core_photon_frequency	<real>	Frequency of photons emitted from core, if using monochromatic emission
core_timescale	<real>	?
core_spectrum_file	<string>	filename of file to read to set spectrum of core emission
core_fix_luminosity	0 = no | 1 = yes	In steady state calculations, will rescale to fix output luminosity

4.4 Multiple Point Sources

CHAPTER 5

Opacity

$$J = \int I_\nu d\nu$$

5.1 Opacity Settings

CHAPTER 6

Hydrodynamics

6.1 Homologous Expansion

6.2 Lagrangian Hydodynamics

CHAPTER 7

Indices and tables

- genindex
- modindex
- search