SPHERLS User Guide 1.0

Chris Geroux

May 23, 2013

0.1 Introduction

This manual is primarily designed to be used to get a user up and running quickly. It also includes some information on how to go about modifying and developing SPHERLS. The reference manuals are useful for determining specific information about variables, functions and classes.

0.1.1 Overview

SPHERLS stands for Stellar Pulsation with a Horizontal Eulerian Radial Lagrangian Scheme. There are three components to SPHERLS: SPHERLS itself which does the hydrodynamics calculations, SPHERLSgen which creates starting models, and SPHERLSanal which is able to manipulate the output files. All three components have their own reference manuals where details of the various class, functions, and variables are described.

0.1.2 The Basics

SPHERLS calculates the radial pulsation motions together with the horizontal convective flow of stars. The radial pulsation can be described by a radial grid velocity, moving the grid inward and outward with the pulsation. The movement of the grid is defined by the motion required to maintaining the mass in a spherical shell through out the calculation. This motion is determined so that it will change the volume of the shell so the newly calculated density when multiplied with the new volume will produce the same shell mass for all times. The total motion of the stellar material is simply the combination of the three velocity components and the grid velocity. The convective motion is the radial velocity minus the grid velocity, combined with the theta and phi velocities. This is because the grid velocity describes the bulk motion of the pulsation so subtracting it out leaves only the convective motions.

SPHERLS solves the normal hydrodynamic equations of, mass, momentum, and energy conservation. The form of the mass equation, momentum conservation, and energy conservation are:

$$\frac{dM}{dt} + \oint_{\mathbb{S}} (\rho\vec{v}) \cdot \hat{n} d\sigma = 0 \tag{1}$$

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = -\frac{1}{\rho}\nabla P + \nabla \cdot \tau - \nabla\phi \tag{2}$$

$$\frac{\partial E}{\partial t} + (\vec{v} \cdot \nabla)E + P\frac{d\mathbb{V}}{dt} = \epsilon - \frac{1}{\rho}\nabla \cdot \ F + \frac{D_t \mathcal{K}^{3/2}}{L} + \frac{1}{\rho}\nabla \cdot \left(\frac{\mu_t}{\rho P_{rt}}\nabla E\right) \tag{3}$$

where τ is the stress tensor for zero bulk viscosity, E is the specific internal energy, $\mathbb{V}$ is the specific volume, and F is the radiative flux. In addition to these conservation equations an equation of state is needed, in this case the OPAL equation of state and opacities, and the Alaxander opacities at low temperatures are used. The equation of state tables are functions of density and temperature, and produce the energy, pressure, opacity, and adiabatic index of the gas for a given temperature and density. In adiabatic calculations, it is also possible to use a γ-law gas equation of state and in that case an initial energy profile must be specified.

The simulation grid is broken up into two main sections, the 1D region towards the center of the star, the multi-dimensional region towards the surface. The inner part of the multi-dimensional region solves all the conservation equations explicitly, in that the new values for the conserved quantities are directly calculated from the information in the previous time step. In the outer parts of the multi-dimensional region the energy conservation equation is calculated semi-implicitly, which means that the new values are dependent on the new values averaged with the old values to correctly time center the equation. This semi-implicit energy conservation equation can be perturbed and linearised producing a set of linear equations the size of the region being solved implicitly. The solution of these linear equations provide corrections for the temperature. The corrected temperature can then be used to solve for a new set of corrections this processes is repeated until the value of the new temperature converges (the size of the corrections is smaller than some specified amount). The equation of state is a function of temperature and not energy which is why the temperature is perturbed and not the energy. This set of equations for the temperature corrections are solved using the PETSC library.

More info to be included here:

- mention that SPHERLS is parallelized
- Different ways in which SPHERLS can be used, 1D,2D,3D, Adiabatic,Non-adiabatic, implicit, debugging options/test
- mention very quickly how SPHERLS can be used (e.g. give very quick example of running SPHERLS)

0.1.3 Program Flow

This should give a rough idea of how SPHERLS works, but should be kept quite high level

- Describe the grids
- The order of calculation
- When parts of the grid are updated
- when models are dumped

0.2 Installing SPHERLS

There are a number of optional and required third party libraries used by SPHERLS. This section will describe how to install these third party libraries as well as SPHERLS. There are three flavours of the SPHERLS distribution package available: one which includes all third party libraries, one which includes only required third party libraries, and one which does not include third party libraries. It is recommended to use the package including all third party libraries. This version ensures the greatest functionality of SPHERLS and also that these libraries are compatible both with SPHERLS and the below installation instructions. Other version of the distribution package can be desirable to reduce download size, if the third party libraries are already installed, (e.g. upgrading SPHERLS) or if you wish to try out newer versions of the third party libraries and do not want to download the additional data.

A few words on installation before we get into the details of the specific third party package installations. In order for the SPHERLS configuration script (needed for installing SPHERLS) to find the required libraries and include files they have to be installed in at least one of the directories that it looks for them. The configuration script looks for the libraries and include files it requires in the following "standard" locations: `/lib`, `/include`, `/usr/lib` `/usr/include`, `/usr/local/lib`, `/usr/local/include`, `/home/-$USER/lib`, and `/home/$USER/include`. If you install the required libraries in places other than these "standard" locations you will have to manually tell the SPHERLS configuration script where to find them. Running `configure -h` will list the available options to tell the script where to find these include files and libraries.

I am going to assume that the user is installing on a Linux system, more over I will be assuming that the Linux distribution follows a Debian like directory structure (many distributions are based on Debian). The install instructions below assume you do not have root access and must do an install to your home directory (a per-user install). The standard install location for per-user level binaries, libraries, include files and documentation (at least far as SPHERLS is concerned) is in `/bin`, `/lib`, `/include`, and `/share`

directories respectively (is just the linux short hand for `/home/$USER`). Be aware that if you have these directories in your home directory already and are not using them as a standard place to install per-user packages you will likely want to rename your pre-existing directories or a bunch of additional files will be added to them from the installations of various packages below. Alternatively, you can install the libraries and binaries to any directory on your machine just by changing `--prefix=/home/$USER/`, mentioned below, to point to where you want to install it. But as mentioned if you install to a non-standard location you will have to tell the SPHERLS configure script where you put things. Also `--prefix` should always be specified as some of these libraries don't install to standard Debian locations on their own, preferring to install into the directory where they are built.

If you have root access and want to install for all users of the current machine you will likely want to install the libraries into `/usr/local` which can be achieved by setting `--prefix=/usr/local` instead of `--prefix=/home/$USER/` used in the below installations and SPHERLS will automatically check this location for the install libraries.

0.2.1 Requirements

SPHERLS requires the below listed libraries to even compile

gcc/g++ and openMPI must have an open MPI aware compiler

PETSc used as the core matrix solver

The below libraries can be optionally installed but will greatly enhance the usefulness of SPHERLS and are greatly recommended

python python scripts are supplied for various analysis and convenience operations. Having python is highly recommended and will make using SPHERLS far more enjoyable. Many SPHERLS python scripts use the following python libraries:

- **numpy** numerical python, fast array operations
- **matplotlib** for creating plots
- **scipy** for interpolating in equation of state and opacity tables when making new files

evtk a python library to make writing vtk files easier

- **Cython** needed to build evtk

fftw3 used for frequency analysis

hdf4 for converting model dumps to hdf4 file format for visualizing

jpeg needed by hdf4 library

zlib needed by hdf4 library

Doxygen used to create documentation from source code via "make docs"

0.2.2 Installing OpenMPI

- Download OpenMPI from the open mpi site
- add library path to `LD_LIBRARY_PATH`
- ./configure –prefix=¡path-to-final-location-of-install¿
- make
- install

0.2.3 Installing PETSC Library

Version `petsc-lite-3.1-p8`, has been tested to work with SPHERLS. `petsc-lite-3.2-p7` is known to be incompatible, which as of this writing is the current version of the PETSc library. At some point in the future support for the newer version of the library maybe added. The below commands will install PETSc into your home directory. ASIDE: I have also had difficulties installing PETSc on Fundy, and Placentia ACENet machines.

- Download PETSc library, from the PETSc website
- Then untar and unzip it with `tar -xzf petsc-lite-3.1-p8.tar`
- To install the library change into the directory made when you extracted the archive and type the following commands:

  1. `./configure PETSC_DIR=$PWD --prefix=/home/$USER/ --with-c++-support --with-c-support --with-shared --download-c-blas-lapack=1 --with-x11=no` where `$USER` is the environment variable corresponding to your username. Note that the `PETSC_DIR=$PWD` needs to come first. Also `--download-f-blas-lapack=1` only works if there is a fortran compiler present, which isn't strictly needed otherwise. Using `--download-c-blas-lapack=1` seems to work in this case

2. `make all` Often at the end of the configuration stage the configuration script will give the command to make the library. One should use this over the above if given.
3. `make install` as with the `make all` the makeFile will also likely tell you the command needed for the installation, which should be used over the one provided here.
4. `make PETSC_DIR=/home/$USER/lib test` will test the code

- You will then need to add the following line to you `.bashrc` file to assure that you will pick up the library `export PETSC_DIR=/home/$USER/lib`

0.2.4 Installing FFTW Library

- Download the FFTW Library from the FFTS website. Version fftw-3.2.2 has been tested to work with SPHERLS.
- Then untar and unzip the downloaded with something like `tar -xzf fftw-3.2.2.tar.gz`
- To install the library change into the directory made when you extracted the archive and type the following commands:
  1. `./configure --prefix=<path-to-final-location-of-library>`
  2. `make`
  3. `make install`

0.2.5 Installing Cython

- `tar -xzf Cython-0.19.1.tar.gz`
- `cd Cython-0.19.1`
- `python setup.py install --prefix=/home/$USER`
- This will require you adding the line `export PYTHONPATH=/home/-$USER/lib/python2.7/site-packages/:$PYTHONPATH` to `.bashrc` so that python can find your installation of Cython, the exact path may depend on your version of python.

0.2.6 Installing evtk

- `tar -xzf evtk.tar.gz`
- `cd evtk`
- `python setup.py install --prefix=/home/$USER`
- This will require you adding the line `export PYTHONPATH=/home/-$USER/lib/python2.7/site-packages/:$PYTHONPATH` to `.bashrc` so that python can find your installation of evtk, the exact path may depend on your version of python also you only need to have this line in your `.bashrc` file once

0.2.7 Installing ffmpeg

useful for making movies from stills

0.2.8 Installing yasm

I think this is needed by ffmpeg

0.2.9 Installing HDF4 Library

Building these libraries requires gfortran and the jpeg library. If using the SPHERLS distribution package including third party optional libraries, version 4.2.8 of the HDF4 library is available in the `libs` subdirectory. Otherwise this library can be downloaded from the hdfgroup website. The most recent version tested to work with SPHERLS is version 4.2.8, version 4.2.7 has also been tested to work. To build the version included with the SPHERLS distribution package use the following set of commands:

- `cd <SPHERLS package directory>/libs`
- `tar -xzf hdf-4.2.8.tar.gz`
- `cd hdf-4.2.8`
- Run `./configure --prefix=<path-to-final-location-of-library> CFLAGS="-fPIC" CXXFLAGS="-fPIC"`.
  If the jpeg libray isn't found, see section in this user guide about installing the jpeg library. Note that for the hdf library the `<path-to-final-location-of` should be set if doing a global install as well as doing a per user install as the default install directory is inside the build directory, which is

not what is usually wanted. So if doing a global install it will probably want to be set to `/usr/local` or for a per-user install `/home/$USER`.

- `make`
- `make install`

The above commands have only been tested with the version including with the SPHERLS distribution package other versions may require slight modifications to the above commands but should be reasonably similar.

0.2.10 Installing Doxygen

The latest version seems to hang while creating documents. doxygen-1.5.6 is known to work.

0.2.11 Installing Python

This could be pain if not simply doing apt-get install

0.2.12 Installing SPHERLS

After all the third party libraries are available this should be easy

0.2.13 .bashrc

What you need to add to .bashrc after all the libraries are installed

0.3 Using SPHERLS

Bits to include in this section are (some of these might be better in a different section, perhaps Developing SPHERLS):

- Generating a starting model
- The XML configuration file
- Starting a calculation
- getting data
- watchzones
- model dumps

- debug information
- post calculation analysis
- python scripts and plotting
- Adiabatic Calculations
- 1D, 2D, and 3D
- γ-law gas
- Sedov Blast wave test
- Non-Adiabatic Calculations
- 1D, 2D, and 3D
- Tabulate EOS
- Different versions of the energy equation
- LES models
- creating a new EOS file using `eos_interp.py`

0.4 Modifing or Developing SPHERLS

This section should include

- Basic layout/design of the code
- model output
- data monitoring
- watch zones
- peak KE tracking
- internal/versus external variables
- message passing
- grid layout
- ranges of grids

- boundary regions
- grid updating
- How to document SPHERLS
- Premade test for SPHERLS after modification
- reference calculations
- restart test
- calculation test (if not modifying calcluation part of SPHERLS)
- How to modify SPHERLS
- Common changes
- How to add a new internal variable

    1. **Add to the internal variable count:** Decide in what cases the variable will be needed, 1D calculations, 2D calculations, when there is a gamma law gas or a tabulated equation of state, adiabatic or non-adiabatic etc. Then once decided it can be added to the total number of internal variables Grid::nNumIntVars by increasing the value by one in the function modelRead in the section below the comment "set number of internal variables ..." under the appropriate if block. If the specific if block for the situation you need isn't there, you can create your own, and add it there.
    2. **Create a new variable ID:** In the grid.h file under the Grid class are variable ID's. These ID's simply indicate the location of the variable in the array. One must add a new ID for the new variable as an integer. The value of the ID is set in the function modelRead in the same section as the number of internal variables. The value used should be the last integer after the last pre-existing variable ID. This should also be Grid::nNumVars + Grid::nNumIntVars -1. The ID should also be initialized to -1, so that the code knows when it isn't being used. This is done in the grid class constructor, Grid::Grid. Simply add a line in the constructor setting your new ID = -1.
    3. **Set variable infos:** Decide what the dimensions of the new variable will be. It can be cell centered or interface centered. It can also be only 1D, 2D, or 3D. Of course it will be only 1D if the

entire calculation is 1D, or 2D if the calculation is 2D, but if the calculation is 3D it could also only be 2D, or 1D, and if 2D it could be only 1D. Also decide if the variable will change with time, dependent variables are only initialized and not updated during the calculations. This information is given to SPHERLS in the setInternalVarInf function in the physEquations.cpp file. The variable that is set is Grid::nVariables. It is a 2D array, the first index corresponds to the particular variable in question, the ID you made in the previous step can be used as the first index of this array. The second index refers to one of the three directions (0-2) or the time dimension (3). If the variable is centered in the grid in direction 0 (r-direction) then this array element should have a value of 0. If the variable is interface centered in the grid in direction 0, then this array element should have a value of 1. If it isn't defined in direction 0 (for example the theta independent variable isn't defined in the 0 direction) then it should be -1. This is the same for the other 2 directions. The last element (3) should be either 0 not updated every time step, or 1 if updated every timestep. There are various sections here which allows one to set variable information based on which conditions are the variable is defined in. Put these variable infos into the most general case in which the variable is defined. At the end of this function variables are automatically adjusted depending on what the number of dimensions the model uses, so this does not need to be considered unless the variable is not used at all for a specific case of dimensions. For example a variable which is defined at cell center for all three cases for the number of dimensions (1D, 2D, 3D) will be automatically adjusted to be not defined in the 3rd direction when only doing 2D calculations, and similarly for 1D only defined in 1st direction and not defined in the 2nd or 3rd directions.

4. **Add functions:** Finally to do anything usefull with your new internal variable functions must be added to initialize the values of the variables, and to update them with time if needed. Initialization functions are called within the initInternalVars function in the physEquations.cpp file. The details of these functions will depend on what the individual variables are intended for. Functions to be called every timestep must be called from the main program loop in the file main.cpp in the appropriate order.

- How to add a new external variable

- How to add a new physics functions
- Function naming conventions
- Grid variables
- indecies and their ranges
- SPHERLS debugging tips

0.4.1 The Equations

I will want to give a detailed description of the equations used (probably copied from my notes wiki) so that the reader can easily see a 1-1 correspondence between the equation and the terms in SPHERLS.

0.4.2 Message Passing

Explain message passing in SPHERLS