RF Tutorial

Rhys Hawkins

January 2014

1 Introduction

This document gives a tutorial introduction to using the RF software.

2 The Tutorial Data

The following files should exist in the data directory:

- RF_obs.dat
- namelist_minimal.nml
- namelist_complete.nml

3 Quick Start

The following command line will run a relatively quick (approximately 1 minute in duration) RF inversion of the tutorial dataset. Note that before running this command you should ensure that the results subdirectory exists and make if it's missing.

```
../rf -d data/RF_obs.dat -p results/ -v
```

It is assumed that you are running the RF application within the source code directory. If this is not the case you may have to adjust some of the paths in the above command line.

```
100% Completed
              Birth      Death      Move     Value Hierarch.
 Propose:      6060       6077      5904      6095      5864
 Accept :       318        312      1989      2370      2552
```

And the following files should be written in the `results` subdirectory:

- credible_max.txt
- credible_min.txt
- mean.txt

- misfit.txt
- parameters.nml
- partitioncount_histogram.txt
- partition_x_hist.txt
- sigma_histogram.txt
- sigma.txt

We will show some simple methods of plotting this data in Section 7.

4 Namelist Files

The recommend method for running the RF suite of applications (serial and MPI version) is to use Fortran Namelist files as inputs. These are simple text files specify the input parameters.

In order to run the same simulation as in the Quick Start section, there is a the `namelist_minimal.nml` file in the `data` subdirectory. This file is reproduced here:

```
&rfsettings
 datafile = 'data/RF_obs.dat',
 outputprefix = 'results/',
 show_progress = 1,
/
```

In order to run the simulation using the namelist file, you can use the following command line:

```
../rf -n data/namelist_minimal.nml
```

Once again please note that this tutorial assumes you are running the RF application within the source tree so it may be necessary to adjust the path to the RF executable, name list file, or to edit paths within the namelist file.

Also note that you are able to specify additional parameters on the command line in addition to the namelist file or override parameters set in the namelist file. For example, if you wanted to run the simulation with more steps without creating another namelist file you could use the following command line:

```
../rf -n data/namelist_minimal.nml -t 20000
```

Command line parameters that are after the namelist file parameter will override any setting from the namelist file.

A comprehensive namelist file with all of the parameters set with comments describing the parameters is in `data/namelist_complete.nml`. It is suggested that you use a copy of this file as a template for your own simulations and adjust input data file locations and the output prefix as appropriate.

5 Input parameters

Here we comprehensively list the parameters available to the RF application. We show the namelist file label and in brackets, the equivalent command line switch.

datafile (-d | --data < file >)

The input data file. This file should consist of space separated x, y coordinates of the RF signal.

outputprefix (-p | --prefix < path >)

Where to write the output files. Be sure to add a trailing "/" if the output is intended to be a directory. For example, if you set output prefix to "output" then the file "outputmean.txt" will be written on successfuly completion. If you set the output prefix to "output/" then the file "mean.txt" will be saved to the output directory. Note that the directory must exist or an error will occur.

burnin (-b | --burnin < int >)

The initial number of iterations to ignore in calculating mean and other statistical results.

total (-t | --total < int >)

The total number of iterations to run. Note that for the MPI version, this number represents the number of iterations run by each process.

maxpartitions (--max-partitions < int >)

The maximum number of partitions (or layers) used to represent the model.

pd (--pd < float >)

In the move process, a partition boundary is moved by a random amount sampled from a Gaussian variate with a zero mean and standard deviation of pd. Lowering this value will increase the Move acceptance rate, however setting this value too low may prolong convergence.

vs_min (--vsmin < float >)

The V_s minimum allowable value in Km/s.

vs_max (--vsmax < float >)

The V_s maximum allowable value in Km/s.

vs_std_value (--vsstd-value < float >)

When perturbing the value of V_s in a Voronoi cell, the new value is obtained by adding a random Gaussian value with a mean of 0 and this standard deviation. As a general rule, increasing this value will lower the acceptance ratio of the propose value operations, and visa-versa. Setting this value too low will result in slow convergence.

vs_std_bd (--vsstd-bd < float >)

When proposing the birth of a new Voronoi cell, the value of the new cell is obtained by adding a random Gaussian value with a mean of 0 and this standard deviation to value of the cell the new cell is birthed within. In the reverse process (the death of a cell) this value is also used in the accept/reject criteria. In most cases this value can be set to the same as the `vs_std_value` parameter but can be lowered further to encourage a higher rate of Births/Deaths.

sigma_min (--sigmamin < float >)

The minimum allowable value of the sigma scaling factor.

sigma_max (--sigmamax < float >)

The maximum allowable value for the sigma scaling value.

sigma_std (--sigmastd < float >)

The standard deviation of the Guassian variate when perturbing the sigma scaling value.

seed (-S | --seed < int >)

The seed for the random number generator.

seed_mult (--seedmult < int >)

For the MPI version of the code, the seed value in each process is set to `seed_base` + `mpi_rank` × `seed_mult`. This options has no effect on the non-MPI version of the code.

show_progress (-v | --verbose)

For the serial code, set this to 1 in the namelist file to show progress information. In the MPI code this option has no effect.

xsamples (--xsamples < int >)

xsamples specifies the number of samples along the x or depth coordinate for output curves such as the mean.

ysamples (--ysamples < int >)

ysamples specifies the number of samples along the y or Vs coordinate and equates to the number of bins in the histogram used to generate the credible intervals.

6 Output

The following files are outputted after a successful completion of a dataset.

credible_max.txt

Credible ranges for the value of Vs versus depth are created and this file contains the 95 % credible range, ie 5 % of values are above this value. The file contains `ysamples` lines each containing `xsamples` V_s values. It is the same format as the `mean.txt` file.

credible_min.txt

This file contains the 5 % credible range, ie 5 % of values were below this value. The file contains `ysamples` lines each containing `xsamples` V_s values. It is the same format as the `mean.txt` file.

mean.txt

The mean profile of Vs as a function of depth from the inversion.

mean.txt

This file contains the mean of the ensemble (n equals total minus burnin iterations). The file contains `ysamples` lines each containing `xsamples` V_s values. The actual coordinate of the individual columns and rows can be determined from the corresponding `xcoords.txt` and `ycoords.txt` files.

misfit.txt

This file contains the misfit of the model to the data as a function of iteration number. It is a useful metric to determine whether the solution is converging and if the burnin parameter is sufficient to prevent the initial "burnin" period from affecting the mean etc.

parameters.nml

This file contains the input parameters used to run this particular RF simulation in the form of a Fortran namelist file. You should be able to run the exact same simulation again using this file as the namelist file input, eg:

```
<path to executable>/rf -n parameters.nml
```

Note that relative paths may need to be adjusted within the namelist file or alternatively RF must be run from the same directory as originally.

partitioncount_histogram.txt

This file contains a histogram of the number of partitions (layers). Column 1 is the number of partitions and column 2 is the count of solutions with this many partitions.

partition_x_hist.txt

This file contains the histogram of the location of partition boundaries and highlights higher probabilities of abrupt changes in Vs in as a function of depth.

sigma_histogram.txt

This file contains a histogram of the sigma hierarchical scaling factor. Column 1 is the centre of the bin value of sigma and Column 2 is the count of values within that histogram bin.

7 Sample Plotting Scripts

In the `scripts` subdirectory, there are a set of python scripts for plotting the results of the tutorial simulation. These scripts use Python 2.x and require the following Python packages:

- matplotlib (Tested with version 1.1.1)
- numpy (Tested with version 1.6.1)

The main script is `plot.py` and this script contains the main plotting routines. The other scripts use functions in the scripts to plot particular results. Note that the scripts expect to be run from within the scripts directory and expect the results in the `../results` directory. A description of each of the plotting scripts follows:

plot_mean_credible_filled.py

Plots the mean and with the credible interval shaded.

plot_mean_credible.py

Plots the mean with credible minimum and credible maximum as dashed lines.

plot_mean.py

Plots the mean curve.

plot_mean_vertical.py

Plots the mean curve with the depth along the vertical axis.

plot_misfit.py

Plots the value of the misfit as a function of iteration number.

plot_partition_count_hist.py

Plots a histogram on the number of partitions or layers used for the model.

plot_partition_x_hist.py

Plots the histogram of the location of partition or layer boundaries.

plot_sigma_histogram.py

Plots the histogram of sigma scaling values.

plot_sigma_history.py

Plots the history of the sigma scaling value as a function of iteration number.

8 MPI Instructions

8.1 Terrawulf

Due to the rf application using some more modern features of the Fortran language, the installed versions of the Fortran compilers fail to compile the application. As a temporary solution, I have locally installed a more up to date version of GNU Fortran and have compiled a version of rf for others to use.

To use this version of rf, you need to adjust your environment as follows (in the second last line the trailing slash is a line continuation, when you input this line the following line should immediately follow the colon without spaces):

```
module load openmpi_144
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$MPI_DIR/lib:
export PATH=$PATH:/home/rhys/install/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/rhys/install/lib: \
  /home/rhys/install/lib64
```

Note that these commands can be added at the end of your `.bashrc` file in your home directory so that rf is available each time you log into Terrawulf.

To test that this is working, you can run rf with no arguments and ensure that you see the following output:

```
> rf_mpi
 A sources file must be specified on the command line.
```

If you receive another error message about missing libraries then please use these diagnostic outputs at the end of this sub-section to try and determine which path is missing/incorrect.

8.1.1 Running

To submit a PBS job, there is a template bash shell script in for submitting on the Terrawulf supercomputer in the `pbs` subdirectory called `pbs_terrawulf.sh`. You can use this script as a template and simply adjust the 2 variables `RF_NAMELIST` and `RESULTS_DIR` with your own file/directory as well as the usual number of CPUs and walltime PBS settings .

```
> qsub pbs_terrawulf.sh
```

If a problem with running the binary is encountered, check the files `pbs_raijin.sh.o<number>` and `pbs_raijin.sh.e<number>` for information. If these do not show any error information then check the `$OUTPUTDIR/mpi.out` file as the rf application may have had trouble loading one or more of the files of there may be a problem with the input parameters.

8.2 NCI Raijin

8.2.1 Compilation/Installation

Compilation under NCI Raijin is relatively straight forward. First compile the RJMCMC library as follows (adjust version number as required, and version of OpenMPI as required):

```
module load openmpi/1.6.3
tar -xzf RJMCMC-1.0.4.tar.gz
cd RJMCMC-1.0.4
./configure --prefix=$HOME/install
make
make install
```

For RF, the compilation steps follow as per usual local install instructions.

```
> module load openmpi/1.6.3
> tar -xzf rf-0.9.1.tar.gz
> cd rf-0.9.1
> export PKG_CONFIG_PATH=$HOME/install/lib/pkgconfig
> ./configure --prefix=$HOME/install
> make
> make install
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/install/lib
```

You should be able to now run the `rf_mpi` application.

```
> $HOME/install/bin/rf_mpi
 A data file must be specifed on the command line.
```

If you receive error messages then please ensure that your `LD_LIBRARY_PATH` is set correctly and you have the correct OpenMPI module loaded.

8.2.2 Running

To submit a PBS job, there is a template bash shell script in for submitting on the NCI Raijin supercomputer in the `pbs` subdirectory called `pbs_raijin.sh`. You can use this script as a template and adjust the variables in this script to point to your own personal directories and input files. To run:

```
> qsub pbs_raijin.sh
```

If a problem with running the binary is encountered, check the files `pbs_raijin.sh.o<number>` and `pbs_raijin.sh.e<number>` for information. If these do not show any error information then check the `$OUTPUTDIR/mpi.out` file as the rf application may have had trouble loading one or more of the files of there may be a problem with the input parameters.