A python code for numerically finding solutions to seismic surface waves

Chirayu Gupta,
3rd year, BS-MS IISER Pune,
Roll No.: 20191060
Supervisor: Arjun Datta

Semester Report

Contents

1 Background **3**
- 1.1 Srgramf . . . 3
- 1.2 Earthsr . . . 3

2 Objectives of the project **4**

3 Previous changes to srgramf **4**

4 Work done on earthsr **4**

5 Work done on srgramf **4**
- 5.1 Used library methods to parse specs/config files . . . 5
- 5.2 Using constants/functions from the library classes . . . 5
- 5.3 Appropriate replacement for common blocks . . . 5
- 5.4 Using OOP principles . . . 6

6 Working of srgramf **6**
- 6.1 Input . . . 6
- 6.2 Output . . . 7
- 6.3 Details of input parameters . . . 7
- 6.4 Outline of the algorithm . . . 7

7 Future plans/Conclusion **7**

1 Background

The earth module consists of 2 modules of seismic code written in Fortran 77 which solves certain problems in seismology regarding solutions of seismic surface waves namely the **Love waves** and the **Rayleigh waves**.

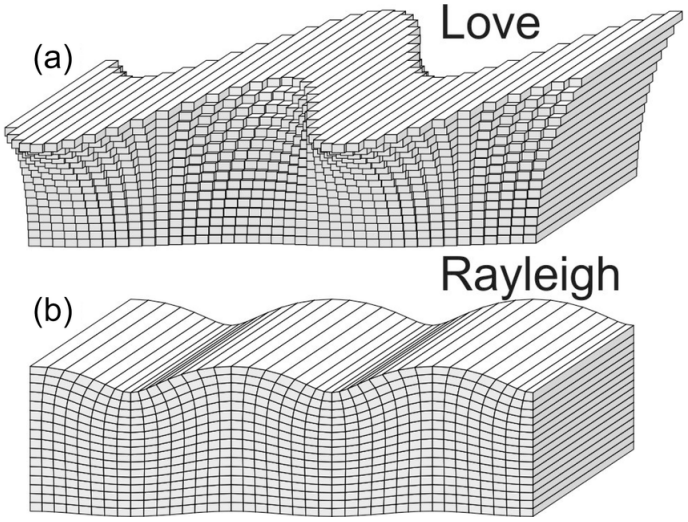

Figure 1: Love (a) and Rayleigh (b) surface-wave displacements for a horizontal propagation from left to right. Love waves are purely transverse motion, whereas Rayleigh waves contain both vertical and radial motion. After Shearer (2009).[3]

Love waves(Figure 1 (a)) have a horizontal motion that moves the surface from side to side perpendicular to the direction the wave is traveling. Of the two surface waves, Love waves move faster.

Rayleigh waves(Figure 1 (b)) cause the ground to shake in an elliptical pattern. This motion is similar to that observed in ocean waves. Of all the seismic waves, Rayleigh waves spread out the most, giving them a long duration on seismograph recordings.[1]

The two modules in earth as mentioned above are:

1.1 Srgramf

Srgramf is a program for computing seismic surface wave Green functions and synthetic seismogram in 1-D Earth models. It is designed to be compatible with the output of earthsr.

1.2 Earthsr

Earthsr is a program to compute Rayleigh and Love wave dispersion curves in a one-d medium for sources and receivers at specified depths. The output of this program is a table of dispersion and excitation information which then is passed to the srgramf program.

2 Objectives of the project

In the summer of 2021, I had worked on translating the earthsr module and had finished translating it to work for Love waves. For the semester project my objectives were:

- Finish the translation of earthsr module for Rayleigh waves.
- Parallelize the translated earthsr code to make its performance comparable to the original Fortran code.
- Translation of the srgramf module into python and make relevant changes to the code to improve performance and readability of the code.

3 Previous changes to srgramf

- Srgramf was initially called **srgram**. Srgram was rewritten to work in the frequency domain(hence the 'f')
- Srgram was a rewrite of **gram** by Steve Roecker to work with earthsr(which was also a rewrite of the original earth code by Steve Roecker) hence the 'sr'

4 Work done on earthsr

The translation work of earthsr was finished during the summer. However certain updates were made to the code to improve performance and readability of the code.

- The whole code was parallelized thus reducing the runtime of the code by a factor of n.
- The procedural like code was broken up into various modules including wrapper classes, helper classes and provided extra provisions in the modification of parameters
- Other minor changes were made as suggested by Dr. Arjun Datta.

5 Work done on srgramf

Entire working version of srgram was translated into python for the semester project. The following changes/additions were made to the code.[1]

[1] Do note that the following points give an outline of the changes make to the original code as it is impossible for me to list all the changes in a report. Apart from the mentioned changes, several other changes to the algorithm were made wherever necessary.

5.1 Used library methods to parse specs/config files

The original code included the whole code for parsing the input specs file which contained certain input parameters to the code. This original method of parsing the input file included ways to make comments in the file so that the parser would completely ignore the comment lines. This was modified to use '.ini' files which has built in functions in python to do the parsing. A number of other options were considered which were rejected due to various reasons.

Type of file considered	**Reason rejected**
.yaml	Need of complex data structures was not required
.json	Data was not being transmitted across a server and format was complex for easy modification
.cfg	Format was similar to .ini and I lacked familiarity with this type of file

In the original code, the parser was programmed in a file called parseprogs_mac.f which contained 254 lines of code and due to this decision, those many lines were omitted in the translated code.

5.2 Using constants/functions from the library classes

The original code contained values for various constants of nature. Since these values were never revised since the writing of the original code in the 1980's, I provided revision using accurate values from the various in built libraries in python. Some of those revisions were like using value of pi from numpy, using revised value of flatness of earth, using radius of earth from astropy, etc.

I also provided alternate methods for doing calculations like finding DCT(Discrete Fourier transforms) using numpy rather than handwritten loops. I also made sure to use numpy arrays and matrices wherever possible. The original Fortran code just used arrays everywhere(including matrices) making it very difficult to understand the code and introducing a lot of loops to iterate over elements. Since numpy uses C/C++ as a backend this also makes the code perform much faster.

5.3 Appropriate replacement for common blocks

Fortran uses common blocks to share data between various files/modules. Since this functionality is not available in python, I used other solutions to replicate this behaviour.

- Some of the variables were declared in a common file and this file was imported in all the files requiring the variables. This was done in accordance of the official python documentation's recommendation on creating global variables.[4]
- Some of the variables were packaged in a class and an object of the class was passed as a parameter. Since user defined objects are mutable, these were passed as

parameters and this method was used mostly in cases where it seemed appropriate to use call by reference

- Some variables were simply passed as parameters to functions to the calling files this was used as a last option and especially when it seemed appropriate to do so.

5.4 Using OOP principles

Python being fairly good in working with classes and objects, I incorporated the use of them wherever necessary to make the code less redundant and modular. For example, consider the code snippet in Figure 5.4 , the code shows a class which takes an angle as a parameter to the constructor and stores calculated values of sin,cos,etc. An object of this class type can then be passed to a function and pre calculated values can be used in places where the function uses these values thereby making the code less redundant. This also makes the code more modular as compared to storing sin,cos values in a separate variable.

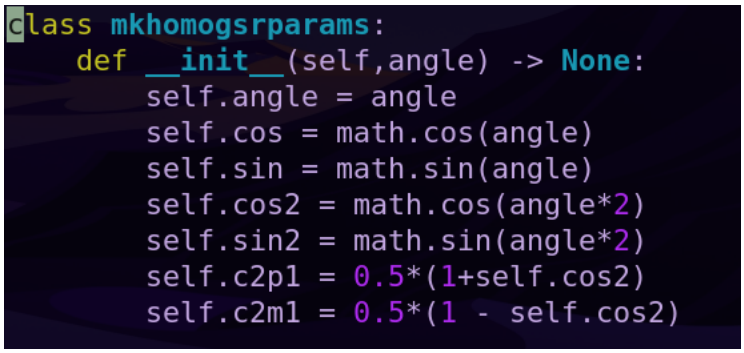

Figure 2: A Code Snippet of the translated code

6 Working of srgramf

srgramf requires a single input file and generates two output files (one ascii and one binary).

6.1 Input

srgram.ini	File containing user specified run parameters
rayfile	Excitation output from earthsr.f for Rayleigh Waves
lovfile	Excitation output from earthsr.f for Love Waves

6.2 Output

grid.<gridpoint>.spec	Binary Greens functions in spectral domain
srgram.log	Ascci Log file with a summary of the run

6.3 Details of input parameters

The file "srgram.ini" contains variables that can be set by the user. Here is a description of the input parameters that can be specified in that file

srgramf is designed to fill up a line with Greens functions at a series of grid points. The endpoints of the line are specified by the variables **glatmin**, **glonmin**, **glatmax**, and **glonmax**. The variable **nx**specifies how many grid points

The variables **evid**, **jy**, **jd**, **jh**, **jm**, **slatd**, **slond** pertain to a given event. **evid** is an event identifier, **jy**, **jd**, **jh**, **jm** are the event origin time, and **slatd**, **slond** are the event location. **d0** is the source depth you want to have between these end points.

rayfile and **lovfile** are the Rayleigh and Love excitation files from earthsr.

mdmin and **mdmx** are the range of modes to include in the computation.

sig, **del**, **gam** are the strike, dip and rake of the mechanism

smoment is the seismic moment in dyne-cm

nz is the z grid number. This is just used for indexing.

df is the frequency interval **nom** Number of frequencies (same as in earthsr)

6.4 Outline of the algorithm

- Surface wave greens function program to be used with a moment tensor as specified for normal modes.
- Works in the frequency domain
- 6 Moment tensor specified in input file or strike/dip/rake to recover source mechanism.
- Algorithm uses dispersion values from earthsr along with momentum tensor and others
- Algorithm works in accordance to equations form mendiguren's paper(1977)
- This algorithm also agrees with Aki and Richards derivation (1981).

7 Future plans/Conclusion

The issue with the current code written in python is that it's slow. Python being a compiled language and not supporting multithreading makes it difficult to improve the performance as well.

1. Link earthsr and srgram into one module.

2. Dockerise the code so that the problem the original code had with dependencies being obselute can be resolved. Of course this can also be done by using virtual environments.

3. Parallelize srgram

4. Performance can be further improved by making the code run on GPU. This can provide a tremendous improvement in performance since a GPU can run upto millions of threads at once.

5. Further improve parallelization of earthsr(Currently parts of code that generate a race condition on parallelizing are not parallized. The use of faster compilers such as pypy3 can also be considered.

References

[1] *Britannica* . URL: https://www.britannica.com/video/181934/rock-vibrations-Earth-earthquake-waves-P-surface.

[2] Arjun Datta. *EC3144 course at IISER Pune.*

[3] *Jens M. Turowski.* URL: https://www.researchgate.net/figure/Love-a-and-Rayleigh-b-surface-wave-displacements-for-a-horizontal-propagation-from_fig3_295863200.

[4] *Official Python Documentation* . URL: https://docs.python.org/3/faq/programming.html#what-are-the-rules-for-local-and-global-variables-in-python.

Code documentations

[5] *Comments of the original code.*

[6] *earth.doc.*

[7] *gram.notes.*

[8] Keith Priestly. *Notes.*

[9] Steve Roecker. *README.example.*

[10] Steve Roecker. *README.srgram.*

[11] Steve Roecker. *README.srgram.*