G_FRAC User Guide

Sabastien Dyer

February 7, 2020

1 Introduction

This document provides the steps for running G_FRAC and how to quickly processes the output on the Compute Canada network of supercomputers. Here are some useful links for getting started with Compute Canada:

- Creating an account
- Documentation
- How to connect to the server through SSH
- How to run a job
- How to transfer files

G_FRAC uses Python3. If you are new to python, a good place to get started is with Pyzo. Some other helpful links for getting started with Python:

- The Python homepage
- Official Python guide for getting started
- Anaconda is a package manager dedicated to data science

2 Setting up G_FRAC

Once you have Python3 you will need to install the required packages. Depending on how you are running python, the command to do this might be different. Typically it is

```
pip install package-name
```

But it can also be "pip3 install" if your system has python2 on it as well. If you are using anaconda, "conda install" works too. Figure out which command to use and install the following packages:

- matplotlib
- easygui*
- pandas
- numpy
- scipy

*Easygui requires tkinter to run. This is automatically included in Python3.7 but must be installed in earlier versions. I believe anaconda includes it as well but I am not certain. It is probably easiest to update to the most recent version rather than figuring out how to get tkinter on your computer.

Once these packages are installed you can run main.py to start the program.

3 Input files

G_FRAC requires several input files formatted in a specific way to run. The program will prompt you for these files as it runs. Examples of these files can be found in the TestData folder.

1. **Garnet Microprobe Data** This is a csv file with five headings: x (mm),Ca,Mg,Fe,Mn. The four components should be given in molar proportion.

2. **Bulk Rock Geochemistry** This is a csv file with the major element geochemistry of the sample you wish to model. It can contain multiple records from different samples. The file should have the header "Name" and all the major oxides analyzed. Possible components are defined in "GeochemConst.py" and can be modified to fit your needs. Reading of headers is case sensitive so make sure you name the headers in the same style as shown in "GeochemConst.py" (e.g. "SiO2" rather than "sio2"). The composition is expected to be weight %.

3. **Blob Data** The program requires the file outputted by Blob3D after processing CT data to isolate garnets in a rock. The file will need to be converted to a xlsx file to be read by G_FRAC.

4 Running the program

1. Run main.py

2. You will be prompted for a csv file for your traverse. This is your microbe garnet data (item 1 in section 3). Navigate to this file and select it.

3. A plot will be displayed showing the compositional profile for all the components. You may click anywhere on the plot to split it in half, the program will ask you to confirm where you want to split it and it will ask you if you want to model growth with the right side or left side, choose your desired option and exit the plot. If you dont like the spot you chose you can click again. If you are only plotting a half traverse (i.e. core to rim), the program will assume that the core is at x = 0.

4. Now you will be prompted for a csv file for your geochemistry (item 2 in section 3). Navigate to this file and select it

5. You will be prompted to choose the sample you want from this file, it will list the "Name" field of every record in the csv file. Choose one and press ok.

6. There will be four boxes to enter input into:

- **Scanned Volume:** This is the volume of the rock in cm^3 that you scanned for analysis using Blob3D. If you only blobbed a subvolume, enter the subvolume instead.
- **Density:** This is the density of the rock you scanned in g/cm^3
- **Database Filename:** This is the filename of the database you want to use for making your phase diagrams in theriak-domino
- **Radius Interval:** The program will model chemical fractionation during growth of garnet for every garnet in your blob file. Every garnet grows at a very small interval and modelling the phase diagram at each interval would be overkill. This field tells the program at what radial growth increments of the biggest garnet it should create output files. If you choose 0.1mm, there will be script files for the rock composition after the biggest garnet has a 0.1mm radius, 0.2mm, 0.3mm etc until the very end. If the biggest garnet's radius is not perfectly divisible by your interval, the last set of output files will represent a smaller interval. See section 7.2 for more details.

7. You will be prompted for the blob file (item 3 in section 3). Navigate to the xlsx file and select it.

8. Now select the folder where you want to save the output. I recommend using an empty folder, it will not warn you if it is overwriting older files.

9. This is a test feature, it will ask you if you want to model as a sphere or ellipsoid. Right now I havent found any difference between the two outputs, I recommend using sphere for now. Once you have chosen one, the model will run.

10. Upon completion you will have your output files, and a plot will be displayed. This plot shows you what the profile that you used for the model looks like after transformation and interpolation. It also displays the increments that the output files were created at. Save this if you wish to keep it.

5 Output

For each radius increment there will be script files produced for the bulk rock composition at that increment. These script files will output a phase diagram, and isopleths of each garnet component. The isopleths plotted have the composition of the next garnet growth shell ±5% and ±10%. There will also be a domjob file with all the script files as well as a file called "sriptList.txt" which is used for running these files using a bash script.

6 Using HPC to make phase diagrams

To be added later

7 What the model is doing

This model is an inverse model which does not actually use any thermodynamics to model growth. The thermodynamics is applied later when running theriak-domino. This is a step by step breakdown of what is actually happening:

7.1 Processing the input

The very first thing that must be done is interpreting the user input. The input files are described in section 3 and the manually entered input is described in section 4. To reiterate, the user is inputting the following:

- Major element composition of the biggest garnet from core to rim or rim to rim
- Bulk rock geochemistry of the rock of interest in weight %
- Size of each garnet in a CT scanned volume of rock
- Total volume of rock scanned
- Density of the scanned rock

These are the items important to actually running the model. Before the model begins, it needs to establish three key parameters:

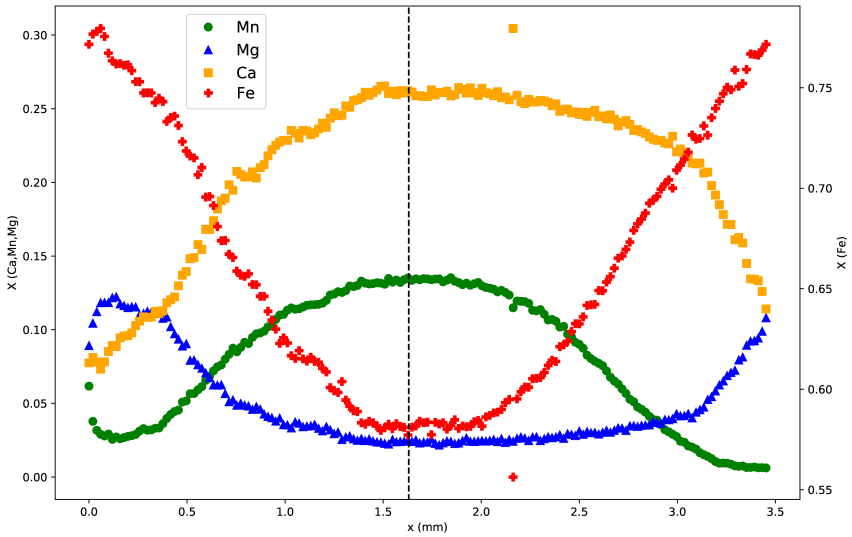
1. The volume and radius of every garnet in the scanned volume
2. The composition of the biggest garnet from core to rim
3. The amount of mols of each component in the scanned volume of rock

Item 1 and 2 on that list are easy to establish. The volume is recorded in the Blob3D output file and can be used to calculate radius, assuming that it is a sphere. The composition of the biggest garnet is given by the user as well. The side of the chemical profile chosen by the user will be extended to fit the radius of the biggest garnet in the Blob3D file. This is accomplished by taking the length of the profile and comparing it to the calculated radius of the biggest garnet. Usually the biggest garnet radius is larger than the length of the split profile, to account for this discrepency, the program will modify the length of the profile to fit the radius of the largest garnet. This is accomplished using one of two possible methods:

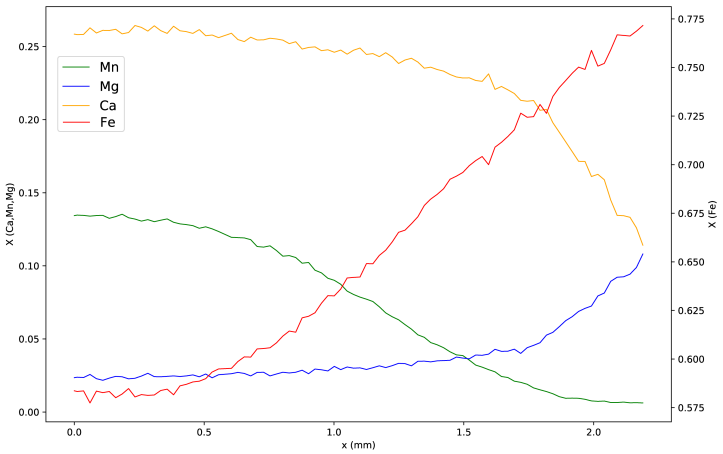
1. The length of the profile is stretched to match the radius of the biggest garnet using the equation: $f = \frac{r_b}{r_p}$ where f is the scaling factor, r_b is the radius of the biggest garnet and r_p is the length of the profile. If x is distance from the garnet core, then every x value in the profile is multiplied by f to rescale
2. The first three points of the traverse are averaged then every x value in the profile is increased by $r_b - r_p$. A new point at $x = 0$ is made with a composition equal to the average that was calculated

Presently, method 1 is used when rescaling a profile that is longer than the biggest garnet and method 2 is used when the profile is shorter. I believe method 2 is a more appropriate than stretching because the most likely reason for a disparity between r_b and r_p is that the profile was not of the biggest garnet but the second or third biggest instead. In this case, it makes more sense that the typically homogeneous core is extended inward, maintaining the length scale at the rim, rather than stretching the distance between every point. If the disparity is a result of non-spherical garnets, then method 1 may be more appropriate.

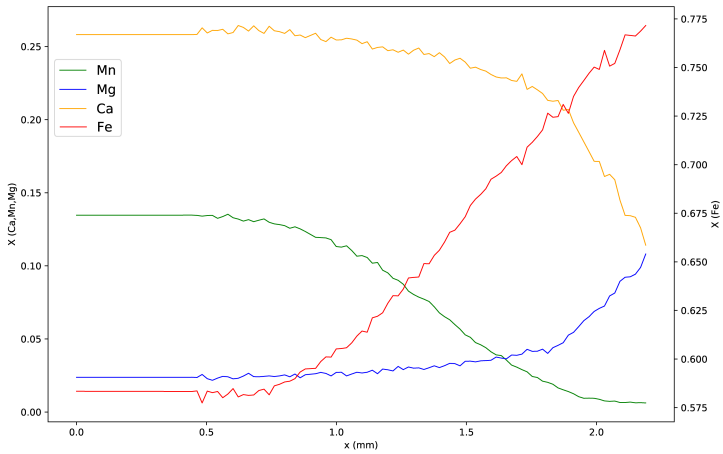
After the profile has been transformed, a linear interpolation is done between every point in the profile. See figure 1 for an example of the transformations and interpolations.



(a) Initial Profile



(b) Stretched



(c) Extrapolated

Figure 1: The right half of the initial profile(1(a)) was shorter than the radius of the biggest garnet by ~ 0.5mm after splitting at the dashed black line. The two possible transformations are shown stretching(1(b)) and extrapolating inwards(1(c))

To calculate the total amount of mols in the system, it requires the mass of the scanned rock and the bulk rock composition. In this case we use density and volume to calculate mass. The weight% provided in the geochemistry data-file can be used to calculate the absolute weight of each component. The moles(n) of each cation is then calculated using the following equations:

$$n_i = z m_i M_i$$

Where z is the number of cations in oxide i, m is the mass fraction of the oxide, M is the molar mass of the oxide. Note that Fe3+ and Fe2+ will both be added together as Fe2+. Oxygen is recalculated assuming reduced oxides. H_2O is assumed to be in excess with 200mol of H and 100mol of O CO_2 is assumed to be reduced and in excess with 100 mol of C.

One additional parameter is hard-coded in "GarnetCSD.py", this is "NUM_SHELLS" and can be changed as desired. It is set to 3000 by default and is used to calculated the radial growth increments (dr). Where $dr = \frac{r_b}{\text{NUM\_SHELLS}}$. The interpolated profile is broken up into 3000 (unless NUM_SHELLS is changed) segments with a width of dr. Each segment has the average composition between the right and left side of the segment.

7.2 Growing garnet

Once the inital conditions are set, the garnet needs to start growing. Growth occurs at a constant radial rate, one layer at a time. Each layer with a thickness of dr. Every garnet that has been nucleated by the program will have a layer added with the same composition. The steps taken in the algorithm are outlined below and illustrated with figure 7.2:

1. Nucleate the first garnet with radius $r = dr$ with the composition of the first segment

2. Add a layer with a thickness of dr to the garnet with the composition of the next segment

3. Continue adding layers, progressing through the segments of the chemical profile until the current radius of the first garnet r_{g1} is equal to the difference between the final radius of the first garnet r_{G1} and second garnet r_{G2} (i.e. $r_{g1} = r_{G1} - r_{G2}$)

4. Nucleate a new garnet with radius $r = dr$ with the same composition of the outermost layer of the other garnets in the system

5. Continue growing garnets and nucleating them once $r_{gi} = r_{Gi} - r_{G(i+1)}$ where i is the smallest garnet present

6. Stop growth once $r_{g1} = r_{G1}$

During this growth period, the program is not actually changing the bulk rock composition. It is simply adding layers to garnets, according to the composition defined by the corresponding segment of the chemical profile. It is however keeping track of the total moles of each component that is being added to garnet. For calculating the amount of mols for component i of p in layer k, $n_{i,k}$ is derived as follows:

$$n_{i,k} = \frac{m_{i,k}}{M_i}$$

$$f_{i,k} = \frac{X_{i,k} M_i}{\sum_{j=1}^{p} X_{j,k} M_j}$$

$$m_{i,k} = f_{i,k} m_k$$

$$n_{i,k} = \frac{f_{i,k} m_k}{M_i}$$

$$n_{i,k} = \frac{X_{i,k} M_i m_k}{M_i \sum_{j=1}^{p} X_{j,k} M_j}$$

$$n_{i,k} = \frac{X_{i,k} m_k}{\sum_{j=1}^{p} X_{j,k} M_j}$$

Where $f_{i,k}$ is the mass fraction of component i in layer k. The calculation of the mass of the layer m_k is done with $m_k = V_k \rho_{grt}$ where ρ_{grt} is the denisty of garnet defined in "GeochemConst.py" as 4.19g/cm^3. Total mols of a component in a single garnet can be easily calculated by summing the layers. Total mols of a component in all garnets can also be calculated easily by summing all the garnets.

The total mols of a component in all garnet can then be removed from the bulk rock composition to determine the amount of material fractionated

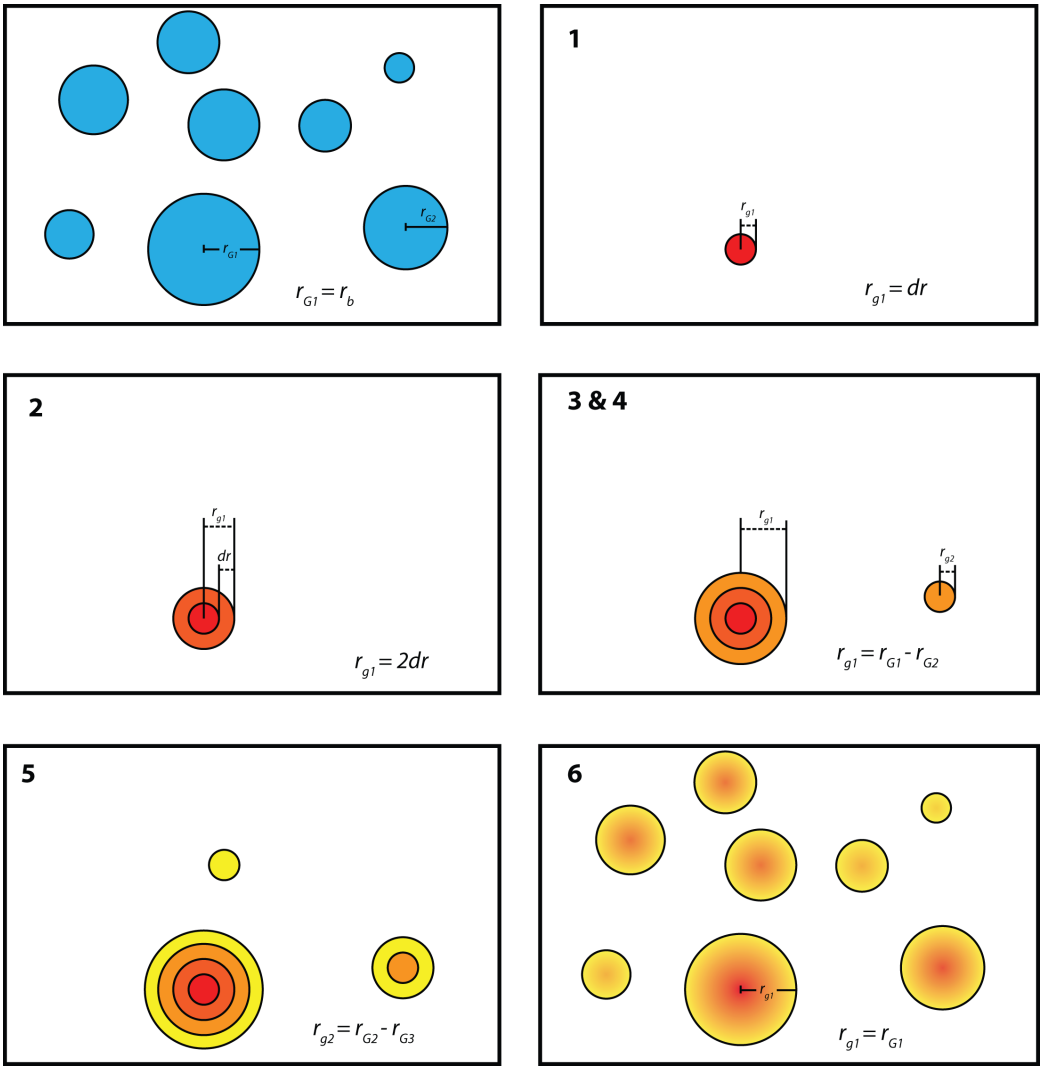


Figure 2: Steps of garnet growth with numbers corresponding to the numerical list in section 7.2

from the system. The program does this calculation at every radius interval defined by the user for creating the output files. It also does it one more time before the final layer is grown.