Homework 3: Modeling of volume change with iron in ferropericlase (Mg,Fe)O
Project implementation
By Luisa Chavarria

The repository is stored in GitHub: https://github.com/luisachavarria2/cmse802_project_hw3.git

1. Introduction

Ferropericlase (Mg,Fe)O is the second most abundant mineral in the Earth's lower mantle. Sodium (Na1+) can incorporate inside the structure through different mechanisms at high pressures conditions (23-136 GPa) (Figure 1). This project looks to understand how substitution affects the dimensions of the unit cell of ferropericlase. Then, it is going to focused on how the volume changes when the concentration of iron changes in the mineral. Linear algebra is going to be implemented to calculate an equation that allows the evaluation of the relation between volume and iron concentration.

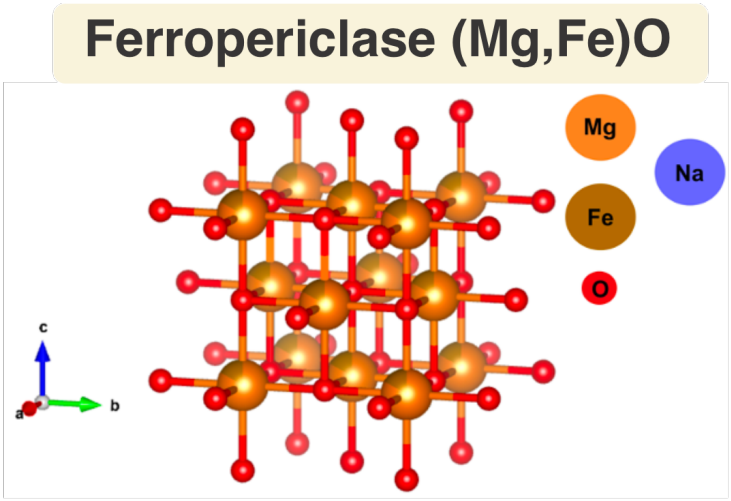

Figure 1. Lattice structure of (Mg,Fe)O with the location of the atoms inside the structure. Sodium (Na) can occupy the positions of Mg or Fe.

Goals:

1. The Homework 3 focuses on the implementationof the project, understanding the data/problem space, establishing testing methods, and implementing core functionality.
2. The initial implementation will focus on the comparison between the least square fit (LSF) and Vegard's law.

My current PhD project studies the incorporation of sodium in ferropericlase, then understanding the initial volumes is important to estimate how sodium changes the structure. In this case, iron is smaller than sodium, but it can be used as an example of how large atoms change the dimensions of the unit cell.

The data that is going to be used in this project has been extracted from https://rruff.info/index.php where different papers have uploaded their XRD data of (Mg,Fe)O at zero pressure and ambient temperature. Currently, the data is limited so creating models for predicting the changes of volume according to the iron concentration is even more relevant.

2. Project Implementation

- **Architecture and component overview**

This is the code structure of the project:

```
cmse802_project_hw3/
├── src/
│   ├── comparison.py
│   ├── data_validator.py
│   ├── fe_calculation.py
│   ├── generator.py
│   ├── linear.py
│   ├── vegards_law.py
│   ├── volume_unitcell.py
│
├── notebooks/
│   ├── hw3_data_analysis.ipynb
│
├── data/
│   └── Fe_volume_fp
├── tests/
│   ├── test_fe_calculation.py
│
├── docs/
│   ├── papers
│   ├── README.md
├── results/
│   ├── report/Hw3_LuisaChavarria_Report
├── README.md
└── requirements.txt
```

- **Implementation status (what's complete and what remains)**

The following Gantt Chart shows the status of the project. The modeling techniques changed compared to the proposed in the Hw2. Instead of running Monte Carlo Simulations, the current project focuses on a linear algebra implementation using Least squared fit and the Vegard's Law to evaluate the change of volume based on the concentration of iron and vice versa.

	Timeline										
Task name	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**	**W13**	**W14**	**W15**
Literature Review											
Cell and Energy calculations											
Monte Carlo Implementation											
Run Simulations and Analyze Trends											
Compare with experimental data											
Finalize Analysis and visualizations											
Final Report and presentation											
Milestones	Homework 1					Homework 2				Homework 3	Project Due

Currently, the model for the least squared fit and the Vegard's law has been developed, as well, as visual comparison of the change of volume with iron concentration. I also tested some data for my PhD project of one of the samples that I have XRD data and want to know the concentration of iron in the sample. So, the next step is to optimize some of the code to make it run more smoothly and reduce redundancy. Also, some additional tests with more samples of ferropericlase to estimate the concentration of iron in the sample are missing.

- **Key algorithms and methods used**

This project uses linear algebra to model the change of concentration of volume as the iron concentration increases in ferropericlase (Mg,Fe)O. The implementation included Least Square Fit (LSF) and the linear model using Vegard's Law for an overdefined system.
The Least Square Fit (LSF) has the form y = c1x + c2. Where:

- y is v and it's linear. It represents the volume of ferropericlase
- x is the concentration of iron in ferropericlase
- c1, c2 are the coefficients we are trying to define. Where c1 is the slope of the fitted line and c2 is the intercept of the line in the y axes

Then, the equation we want to fit is:

$$V = c1x + c2$$

On the other hand, the Vegard's law states that the lattice parameter (or molar volume) of a solid solution varies linearly with the concentration of its constituents.

$$a_{\mathrm{A}_{(1-x)}\mathrm{B}_x} = (1-x)\, a_{\mathrm{A}} + x\, a_{\mathrm{B}}$$

Where a is the lattice parameter, A and B are the constituents of the solid solution, and x is the fraction concentration. Then, applied to the system (Mg,Fe)O, this would be the model that is going to be implemented in this project using the volume (V) instead of the lattice parameter (a):

$$V(x) = (1-x)V_{\mathrm{MgO}} + xV_{\mathrm{FeO}}$$

Where:
V(x) = molar volume of $(Mg_{1-x}Fe_x)O$
V_{MgO} = molar volume of end member MgO

V_{FeO} = molar volume of end member FeO
x = mole fraction of Fe in the solid solution

Vegard's Law is useful to obtain a rough estimate of the composition of a solution where the experimental data are not available. The expectation is that the solid solution behaves linearly, and the deviations are caused by site ordering, inhomogeneities, and changes in oxidation state.
Then, the least squared fit and the Vegard's law are expected to be similar for an ideally mixed solution.
The calculation of the iron concentration based on the volume used the equation of volume and solved for x in the LSF section. Then:

$$x = \frac{V - c2}{c1}$$

- **Technical challenges and solutions**

*The comparison module was challenging to code because it required to duplicate some of the code that was previously generated for the Vegard's law and the LSF modules. So, one option was to simplify it and include it in a new file script without calling those modules. For the final project, I'm going to look for one option of avoiding duplicates.
*Another challenge was to generate the visualizations of the volume of the unit cell based on the iron concentration. That section required several trial and error to find the right delta for the modeling of the volume. For the final project, I'm going to double check the model to be sure that the values are accurate.

- **Any changes from the original plan in HW2**

I identified a bottleneck in the project that I proposed for the Hw2. In that project, the goal was to calculate how the unit cell changes when sodium was replacing Fe in (Mg,Fe)O. However, it was difficult to find data to test and verify that the results that were provided in the code were accurate. The study of sodium in ferropericlase is part of my PhD project, and currently, there are very few studies where sodium has been analyzed in this mineral. Then, I decided to take a step back and focus on the modeling of the volume of ferropericlase at different concentrations of iron. The results are going to be useful for comparing with the volumes of ferropericlase with sodium incorporated.

3. Technical Optimization

- **Performance bottlenecks identified**

The module comparison.py still has some potential for improving performance using a more simplify code and avoiding the duplication of the fittings methods.

- **Optimization techniques implemented**

The method curve_fit from scipy.optimize was used in the module linear_lsf.py to optimize the calculation of the linear model.

```
"""
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def linear_model(x, a, b):
    """Linear model: V = a * x + b"""
    return a * x + b

def fit_and_plot_linear_model(x, V):
    """
    Fit a linear model to the data and plot it.

    Parameters:
    - x (np.ndarray): Fe mole fraction
    - V (np.ndarray): Molar volume

    Returns:
    - tuple: (a, b) Fitted parameters of the linear model
    """
    try:
        params, covariance = curve_fit(linear_model, x, V, p0=[1, 74])
        a, b = params
    except Exception as e:
        print(f"An error occurred during curve fitting: {e}")
        return None, None
```

Additionally, the scripts were divided into modules that allows optimize the running time and an easier implementation and reusability of the codes.

- **Error handling and robustness measures**

Most of the codes include error handling sections were errors are raised if there is an inconsistency or there are exceptions. For example, the data_validator.py file, includes the function check_data_validity. This function evaluates the quality of the data that is uploaded for this project and it raises ValueErrors if x and V are empty, contain NaN values, have different lengths or infinite values.

```
def check_data_validity(x, V):
    """
    Validates that x and V are suitable for analysis and prints statistics.

    Parameters:
    - x (np.ndarray): Array of Fe mole fractions
    - V (np.ndarray): Array of corresponding molar volumes

    Raises:
    - ValueError: If any validation check fails
    """
    if len(x) != len(V):
        raise ValueError("The length of x and V arrays must be the same.")
    if len(x) == 0 or len(V) == 0:
        raise ValueError("x and V arrays must not be empty.")
    if np.any(np.isnan(x)) or np.any(np.isnan(V)):
        raise ValueError("x and V arrays must not contain NaN values.")
    if np.any(np.isinf(x)) or np.any(np.isinf(V)):
        raise ValueError("x and V arrays must not contain infinite values.")
```

4. Validation and Testing

- **Validation methodology**

The modeling will require test for the following modules:

1. Linear_lsf.py
2. Vegard's_law

3. fe_calculation.py
4. volume_unitcell.py

However, I'm going to show the implementation of the test of the iron calculation in the file fe_calculation.py. Unittest was used for the testing.

- **Test cases and coverage**

Test Case 1. fe_calculation
Test objective: verify that the function estimate_fe estimates the concentration of iron correctly based on the linear equation:

$$x = \frac{V - 75.074}{4.816}$$

where volume is the input in A^3 and x is the output (Fe concentration).

Inputs:
There are 8 test cases where the input corresponds to 0%, 50%, and 100% concentration of Fe. Also, there are some cases where there volume is below the minimum volume, there is invalid data like strings, or missing input. Also, values that are very large or small, or are negative.

Expected output:
For example, for the Test case 5 where the input is none, the output is an exception raised where there is a TypeError indicating that the input is incorrect.

Test steps:

1. Prepare the input value
2. Call the function "estimate_fe(volume)
3. Verify the output and check the returned float value
4. Handle the edges and exceptions

Testing implementation:
The script test_fe_calculation.py can be executed in the terminal using the following command:

- python -m unittest test_fe_calculation.py

This is how the test looks like:
The test includes 6 functions:

- test_zero_concentration : zero Fe concentration (MgO)
- test_half_concentration: 50% Fe concentration (Mg,Fe)O
- test_full_concentration: 100% Fe concentration (FeO)
- test_negative_concentration
- test_invalid_input
- test_print_output

```python
from fe_calculation import estimate_fe

class TestEstimateFe(unittest.TestCase):

    def test_zero_concentration(self):
        self.assertAlmostEqual(estimate_fe(75.074), 0.0, places=5)

    def test_half_concentration(self):
        # x = 0.5 => V = 4.816*0.5 + 75.074 = 77.482
        self.assertAlmostEqual(estimate_fe(77.482), 0.5, places=5)

    def test_full_concentration(self):
        # x = 1.0 => V = 4.816*1.0 + 75.074 = 79.89
        self.assertAlmostEqual(estimate_fe(79.89), 1.0, places=5)

    def test_negative_concentration(self):
        # Should work, but result might be negative
        result = estimate_fe(70)
        self.assertLess(result, 0)

    def test_invalid_input(self):
        with self.assertRaises(TypeError):
            estimate_fe("not a float")

    def test_print_output(self):
        captured_output = StringIO()
        sys.stdout = captured_output
        estimate_fe(77.482)
        sys.stdout = sys.__stdout__
        self.assertIn("estimated Fe concentration", captured_output.getvalue())
```

- **Quantitative validation results**

As an example, I run the following code:
python -m unittest test_fe_calculation.TestEstimateFe.test_zero_concentration
and obtained the correct answer for the test case where the volume is 75.07 A^3 and then the iron concentration is zero.

```
(base) luisachavarria@Luisas-MacBook-Pro-2 hw3 % python -m unittest test_fe_calculation.TestEstimateFe.test_zero_concentration

For the volume = 75.07 Å³ the estimated Fe concentration is  0.00%
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
(base) luisachavarria@Luisas-MacBook-Pro-2 hw3 %
```

- **Analysis of accuracy and reliability**

To analyze the accuracy of the least squared it was compared the LSF vs the Vegard's Law and it was found that both of them are overlapping. Then, this means that it is reliable to use the LSF model to calculate the concentration of iron and to estimate variations of volume based on iron.

5. Results and Discussion

The LSF gave as result the linear fit equation where V= 4.816x + 75.074. The following figure 2 shows the scattered experimental data and the line fitted.

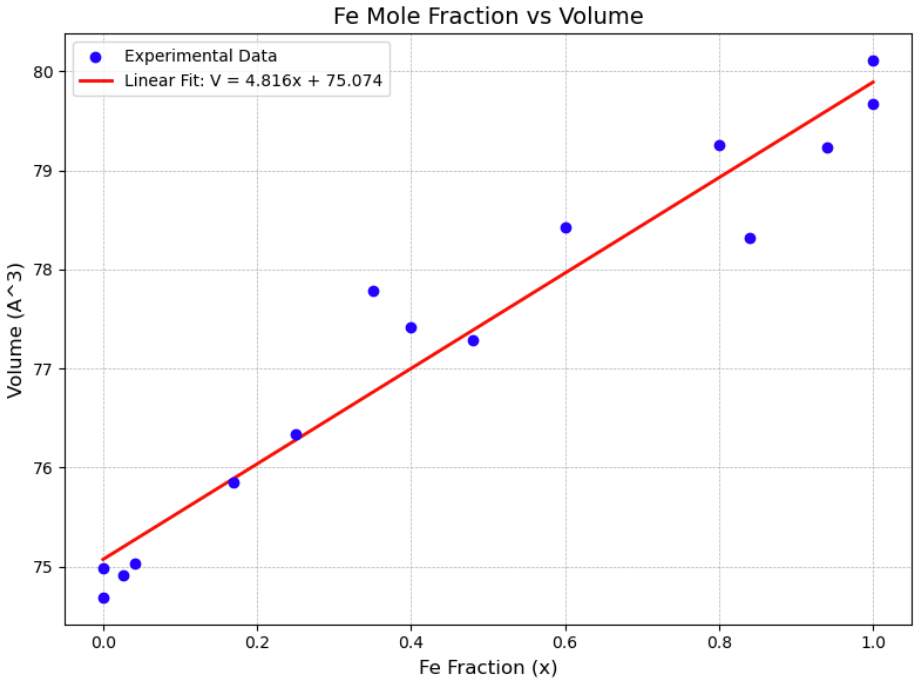

Figure 2. LSF model plotted with the linear fit and the experimental data

The implementation of the Vegard's Law also gave as a resulted an equation with a fitted line. It was found that the equation is similar to the one for the LSF. For the final project, it is going to be evaluated if this is correct as it is expected that the Vegard's law is constrained by the maximum and minimum concentrations of Fe, then the values expected could depend of those values and could differ from the LSF model. The following figure 3 shows plot with the fitted line of the Vegard's Law.

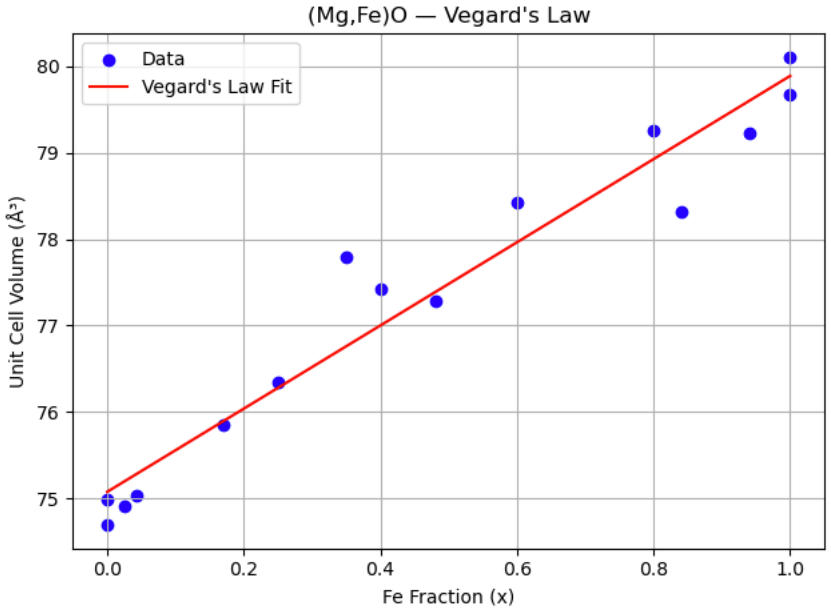

Figure 3. Vegard's law model plotted with the linear fit and the experimental data

It was found that the Vegard's Law and the Least Squares Fit are equivalent. The following figure 4 shows the relation:

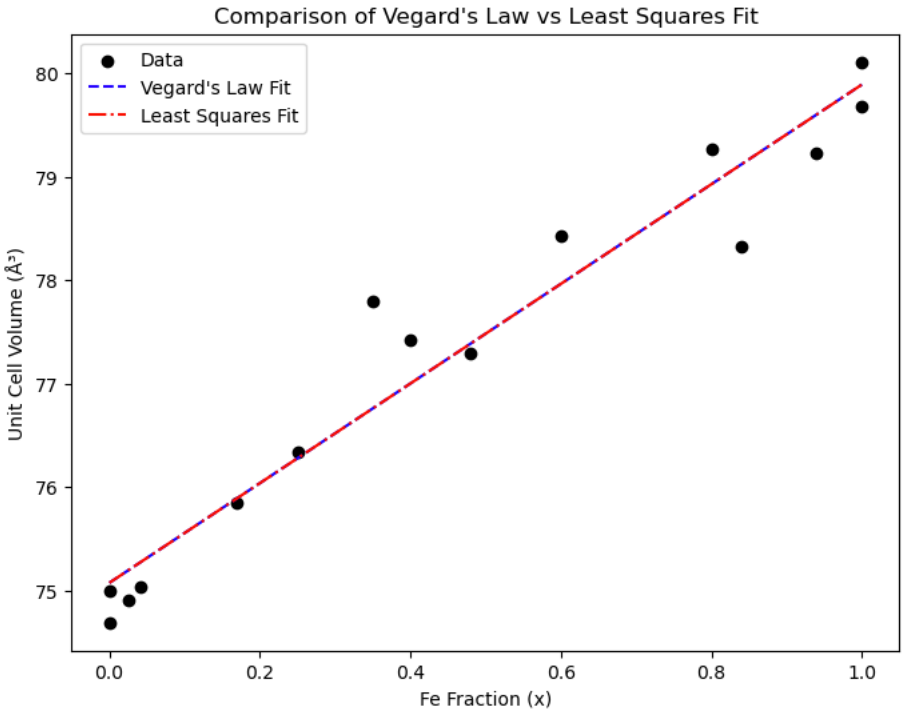

Figure 4. Vegard's law model vs LSF plotted with the linear fit and the experimental data

The modeling of the volume and iron concentrations are useful to estimate the change of data that is not accessible experimentally. For example, in the following figure (Figure 5), there are five models of how the unit cell dimensions change according to the iron and volume. It is clear that as iron is a bigger atom than magnesium, the size of the unit cell is expanding as the concentration increases.

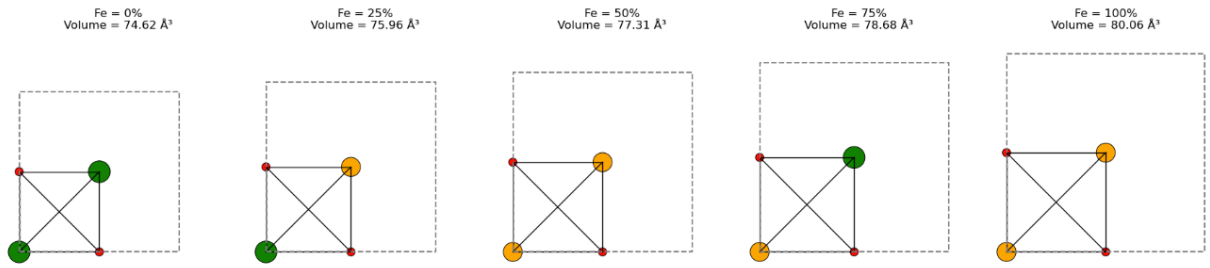

Figure 4. Model of how the unit cell changes with the concentration of iron.

This is relevant to understand substitution and in the future to understand how the incorporation of elements like sodium can change the dimensions of the unit cell in ferropericlase.

Future improvements include to update the code of the unit cell changes to accurately show how the distribution of atoms changes according to the concentration of iron. Currently, the colors green (Mg) and yellow (Fe) are not accurately representing the proportion of iron.

5. Conclusion

- **Summary of achievements**

* It was completed the model for LSF and Vegard's fitting of the experimental data
* It was possible to compare the models and realize that it is accurate to use the LSF for modeling iron concentrations and volumes.

- **Evaluation of approach**

*Some samples that belong to my PhD research has been used to evaluate what model would be useful for the characterization of the XRD data that was acquired previously. So, using these models is helpful for the fully characterization of the iron concentration in the samples
*Using LSF and Vegard's law makes it more robust

- **Next steps for completion**

  * Next steps includes to improve the concentration of iron model and also to add some of my data to evaluate the accuracy of the model

7. References

- Citations for any methods, libraries, or sources used

1. OpenAI. ChatGPT. April 16, 2025. OpenAI. https://chat.openai.com. (Planning and coding of the project)
2. The data from the experimental data was obtained from https://rruff.info/index.php