Phase Field Tutorial

Mathematics 2: Numerical Methods with Python Implementation

Zihang Wang

Central South University

November 5, 2024

Quick Review

What have we got in the last tutorial?

- Intro to PF
- Taylor Formula
- Extrema with Constraint
- Variational Derivative
- Vector Calculus

Great, let's move on.

Numerical Method

By now, you should have acquired the basic knowledge about phase field method. But how to carry out a simulation?

Numerical Method

By now, you should have acquired the basic knowledge about phase field method. But how to carry out a simulation?

The answer is: *Numerical analysis*, or *numerical methods*. That means breaking the mathematical problem into pieces and converting to computer programming problems, then using programming to solve them.

Numerical Method

By now, you should have acquired the basic knowledge about phase field method. But how to carry out a simulation?

The answer is: *Numerical analysis*, or *numerical methods*. That means breaking the mathematical problem into pieces and converting to computer programming problems, then using programming to solve them.

So, in today's tutorial, we are going to cover some numerical methods, together with how to implement them with a programming language. Here we choose Python to implement these algorithms [1]. Don't worry, we will also cover some points about python, start from scratch.

[1] While, next tutorial will focusing on C++, a faster but more complex programming language

What Questions Are We Facing?

As you can see, numerical analysis is a general concept, and for a particular problem, there should be a particular method, or say, algorithm, to solve it. And to find the proper methods to solve our questions, we should analysis the questions we are facing.

Let's look back at the two governing questions we mentioned before:

$$\frac{\partial c_i}{\partial t} = \nabla \cdot M_{ij} \nabla \frac{\delta F}{\delta c_j\left(r, t\right)} \qquad \text{(Cahn-Hilliard)}$$

$$\frac{\partial \eta_p}{\partial t} = -L_{pq} \frac{\delta F}{\delta \eta_q\left(r, t\right)} \qquad \text{(Allen-Cahn)}$$

So, the main question is, how to utilise these equations to evolute the field parameter as we want?

Break Down the Questions

By analysising these two equations (don't forget the exercises in the last tutorial), the following questions should be solved:

Break Down the Questions

By analysising these two equations (don't forget the exercises in the last tutorial), the following questions should be solved:

- How to solve *ordinary differential equations* and *partial differential equations* (usually refered as *ODE* and *PDE*)?
- How to *Integrate*?
- How to deal with ∇? (that is, how to *taking derivative*)
- How to (, if you have seen more about vector analysis,) deal with ∇^2(*laplacian*, sometimes denoted as Δ)?

Break Down the Questions

By analysising these two equations (don't forget the exercises in the last tutorial), the following questions should be solved:

- How to solve *ordinary differential equations* and *partial differential equations* (usually refered as *ODE* and *PDE*)?
- How to *Integrate*?
- How to deal with ∇? (that is, how to *taking derivative*)
- How to (, if you have seen more about vector analysis,) deal with ∇^2(*laplacian*, sometimes denoted as Δ)?

With these sub-questions solved, and a little bit of code, we should be able to, finally, start a (simple) simulation by ourselves, independently.

How Numerical Methods Work?

One of the most obvious features of numerical methods is it use ‘discrete’ to replace ‘continuous’. For example, taking derivative can be approximated by small value interval divided by small variable interval, and integrate can be approximated by sum of small variable interval times its corresponding mean value. Or, to make it clearer, in analytical method, you taking limit to achieve ‘continuous’, while in numerical method, you taking small interval to mimic limit.

How Numerical Methods Work?

One of the most obvious features of numerical methods is it use 'discrete' to replace 'continuous'. For example, taking derivative can be approximated by small value interval divided by small variable interval, and integrate can be approximated by sum of small variable interval times its corresponding mean value. Or, to make it clearer, in analytical method, you taking limit to achieve 'continuous', while in numerical method, you taking small interval to mimic limit.

By now you may have gotten idea of how to solve the sub-questions we mentioned above. If so, all you need is a great tool to check your idea. Here it comes: Python.

What's Python?

Python is a kind of snake, and in the mean time, one of the most popular programming language in the world. Easy to learn, friendly grammar, and most important, its large community, together with hundreds and thousands of great tutorials are Python's outstanding features.

What's Python?

Python is a kind of snake, and in the mean time, one of the most popular programming language in the world. Easy to learn, friendly grammar, and most important, its large community, together with hundreds and thousands of great tutorials are Python's outstanding features.

Let's start from the scratch, that is: download the interpreter first[2].

[2] What is interpreter? While, loosely speaking, it's a program that interprets code to computer and let computer to execute it. It differs from compiler (used by C/C++) that interpreter translates and executes code line by line, while compiler must compile the code to binary file (usually, on Windows, an *exe* file) and then execute it

Let's start with Python

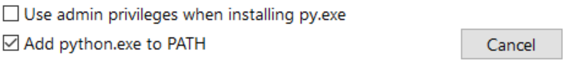
Let's suppose your operating system is Windows[3]. Head to Python's download page , hit the 'Download' button, waiting for the downloading to complete, then you will get the installer. Install it, but remember,

[3]For users of *nix users, things are easier that just install it with the package manager you have (or like)

Let's start with Python

Let's suppose your operating system is Windows[3]. Head to Python's download page , hit the 'Download' button, waiting for the downloading to complete, then you will get the installer. Install it, but remember,

TOGGLE ADD TO THE PATH CHECKBOX!

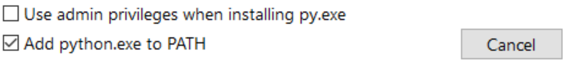


[3]For users of *nix users, things are easier that just install it with the package manager you have (or like)

Let's start with Python

Let's suppose your operating system is Windows[3]. Head to Python's download page , hit the 'Download' button, waiting for the downloading to complete, then you will get the installer. Install it, but remember,

TOGGLE ADD TO THE PATH CHECKBOX!



Then you can just accept the default settings, and after installing, you are ready for coding with Python [4].

[3] For users of *nix users, things are easier that just install it with the package manager you have (or like)

[4] Or, you can using *Microsoft Store* to download Python interpreter, or using package manager to achieve this.

Check your installation

You must can't wait coding with Python, but before that, please check wheather you have installed your interpreter successfully:

1. Open a shell (cmd or powershell, you can use `win + R` to open 'run' and input `cmd` (if you prefer powershell, `powershell`))
2. Input `python --version` and hit `Enter`.
3. If you installed successfully, you will get the version information about python.
4. While, if you get something like '*XXXX is not reconised as xxx*', that error message indicates that you might forget to *add python.exe to PATH*. Re-install or add to PATH manually.

Begin your first Python code

Now, please create a text file, modify its extension name (if you didn't see anything with `.txt`, open extension name in file explorer options) to `py`, and open it. Then write the following code:

```
print("Hello Python World!")
```

Save it, which is your first python script. Then open the shell you like, call python interpreter with your script's path as argument:

```
python C:\path\to\your\python\script
```

Remember to replace the `C:\path\to\your\python\script` *to your script's path.* You should see 'Hello Python World' was printed in your shell.

Edit in Plain Text?

Congrats, you should have made your first step in Python programming. But wait, shall one edit the python script in plain text editor? The answer is: no, of course not. What you need is a modern editor to handle it.

Here I recommend Visual Studio Code with Python extension, and run or edit Python with Jupyter Notebook. You can go to my Github repository to download the instruction about how to set up a Python environment with VS Code. There will also be a repository contains today's code and other resources. Comments and suggestions about these documents are welcomed.

Of course, developing large-scaled Python program usually use *Integrated Development Environment*, *IDE*. The most famous one might be PyCharm by *JetBrains*. Try it if you want.

A Python Tutorial?

By far, we are all talking about Python. One might complain that: ‘Is this a Python tutorial?’

A Python Tutorial?

By far, we are all talking about Python. One might complain that: ‘Is this a Python tutorial?’

We will pause for a while. For thouse who are completely new to Python, the following part is focusing on algorithm, and we will turn to Python in the time we need to implement these algorithms. If you need aids with Python language, there are so many awesome tutorials about Python, from beginer to proficient. If you’d like to learn more about Python, please head for these resources.

A Python Tutorial?

By far, we are all talking about Python. One might complain that: ‘Is this a Python tutorial?’

We will pause for a while. For thouse who are completely new to Python, the following part is focusing on algorithm, and we will turn to Python in the time we need to implement these algorithms. If you need aids with Python language, there are so many awesome tutorials about Python, from beginer to proficient. If you’d like to learn more about Python, please head for these resources.

Now, we are prepared for Python programming, meaning that we are ready to write Python scripts and implement our algorithms with Python (although you might be not familiar with Python yet).

What are ODE and PDE?

ODE and PDE are equations that the solutions are a function(precisely, a family of functions). As their name indicates, ODE contains total derivative, while PDE contains partial derivative. Let's give the definitions for these two concepts:

Ordinary (Partial) Differential Equation

Given F, a function of x, y, and derivatives of y, the equation of the form

$$F(x, y, y', \ldots, y^{(n)}) = 0$$

is called an ordinary differential equation (partial differential equation, if derivatives stands for partial derivative). Then, a n-times differentiable function u satisfies this equation is thus a solution to this ODE (or PDE).

Analytical Solutions vs. Numerical Solutions

You might want to get an analytical solution of a ODE or PDE. Usually it's possible only in homework, and impractical when facing a little bit more complicated equation.

Analytical solution provides to some extent perfect solution to a equation, while in practice, especially in physical or material context, analytical result is 'way more precise' than actual needs, and numerical result can provide a balance between less time consuming with lower precision. And sometimes, numerical method is the only way to solve a over-complicated differential equation.

Analytical Solutions vs. Numerical Solutions

You might want to get an analytical solution of a ODE or PDE. Usually it's possible only in homework, and impractical when facing a little bit more complicated equation.

Analytical solution provides to some extent perfect solution to a equation, while in practice, especially in physical or material context, analytical result is 'way more precise' than actual needs, and numerical result can provide a balance between less time consuming with lower precision. And sometimes, numerical method is the only way to solve a over-complicated differential equation.

So, let's head for the numerical method of solving differential equations. Here we are going to introduce *forward (explicit) Euler method* and *backward (implicit) Euler method*, included in *finite difference method (FDM)*.

Difference Quotient

Consider a function's Taylor expansion at point (denote the interval as Δx):

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \cdots + \frac{f^n(x)}{n!}(\Delta x)^n + r_n(x; x+\Delta x),$$

Now, we want to get approximation of first order differentials, $\tilde{f}'$. To achieve that, take the higher terms to be reminder and eliminate the reminder, and you will get:

$$\tilde{f}'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

That's so called *first order difference quotient*. If you need higher order one, you can calculate it recursively, by taking nth step result as $n+1$th step's input.

Difference Quotient

Consider a function's Taylor expansion at point (denote the interval as Δx):

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \cdots + \frac{f^n(x)}{n!}(\Delta x)^n + r_n(x; x+\Delta x),$$

Now, we want to get approximation of first order differentials, $\tilde{f}'$. To achieve that, take the higher terms to be reminder and eliminate the reminder, and you will get:

$$\tilde{f}'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

That's so called *first order difference quotient*. If you need higher order one, you can calculate it recursively, by taking nth step result as $n+1$th step's input.

Now it's time to try a simple example:

Euler Method from Simple Example

Consider a simple differential equation as follows:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y),$$

and suppose you have already know the start point $(0, u(0))$ of the graph of the solution function u [5]. By subsituting derivative with difference quotient, and labelling the value of the solution function u at each divided points as $u_1 = u(0 + \Delta t)$, $u_2 = u(0 + 2\Delta t)$,...

[5] With initial condition, this problem belongs to initial value problems (IVP).

Euler Method from Simple Example

Consider a simple differential equation as follows:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y),$$

and suppose you have already know the start point $(0, u(0))$ of the graph of the solution function u [5]. By subsituting derivative with difference quotient, and labelling the value of the solution function u at each divided points as $u_1 = u(0 + \Delta t)$, $u_2 = u(0 + 2\Delta t)$,...

Now this differential equation can be written as:

$$\frac{u_n - u_{n-1}}{\Delta t} = f(t, \underline{u}).$$

[5] With initial condition, this problem belongs to initial value problems (IVP).

Explicit & Implicit Euler Method

Here is a underlined notation, $\underline{u}$. That should be the value of u at the point t. What should it be?

Explicit & Implicit Euler Method

Here is a underlined notation, $\underline{u}$. That should be the value of u at the point t. What should it be?

If you choose u_{n-1} as $\underline{u}$, then you get *forward Euler method*. Just multiply Δt to the RHS, then move u_{n-1} to the RHS, you get the formula to derive the value of u from $t = 0$ to wherever you want. Quite easy, right?

Explicit & Implicit Euler Method

Here is a underlined notation, $\underline{u}$. That should be the value of u at the point t. What should it be?

If you choose u_{n-1} as $\underline{u}$, then you get *forward Euler method*. Just multiply Δt to the RHS, then move u_{n-1} to the RHS, you get the formula to derive the value of u from $t = 0$ to wherever you want. Quite easy, right?

If you choose u_n as $\underline{u}$, then you get *backward Euler method*. You must have seen that in this case, you can't directly solve u_n instantly. That's why it is also refered as *implicit Euler method*, and another one method also named *explicit Euler method*.

Explicit & Implicit Euler Method

Here is a underlined notation, $\underline{u}$. That should be the value of u at the point t. What should it be?

If you choose u_{n-1} as $\underline{u}$, then you get *forward Euler method*. Just multiply Δt to the RHS, then move u_{n-1} to the RHS, you get the formula to derive the value of u from $t = 0$ to wherever you want. Quite easy, right?

If you choose u_n as $\underline{u}$, then you get *backward Euler method*. You must have seen that in this case, you can't directly solve u_n instantly. That's why it is also refered as *implicit Euler method*, and another one method also named *explicit Euler method*.

Well, what if choosing other values?

Do it with Python!

Believing this is the time to implement this algorithm with Python, please try it yourself.

You will get hints from the given resources.

1. Review & Intro
2. Preliminary
3. Numerical Methods
   - ODE&PDE
   - Integrate
   - ∇ & ∇^2
4. Summary

Sum It Like Riemann, in Python

You have already know how to approximate a function's derivative. Just imagine how to get approximation of integral.

Sum It Like Riemann, in Python

You have already know how to approximate a function's derivative. Just imagine how to get approximation of integral.

So remind yourself with Riemann integral's definition. Dividing function's domain into small pieces, then times them with corresponding function value, and sum them up. If you continue to take limit of the domain pieces, you get Riemann integral[6]. But if you stop here, you get a approximation of (Riemann) integral.

[6]Actually, Riemann integral's definition is more general that, the domain's division is arbitrary, any division should give the same results, and so on. We don't cover these here

OOP, and other algorithm

Integrate single variable functions using plain Riemann or Darboux summation, should be somehow a simple thing. So we'll try to integrate double variable functions. This brings a problem: How can I define a double variable function? Here we introduce: *OOP, Object Oriented Programming*.

OOP, and other algorithm

Integrate single variable functions using plain Riemann or Darboux summation, should be somehow a simple thing. So we'll try to integrate double variable functions. This brings a problem: How can I define a double variable function? Here we introduce: *OOP*, *Object Oriented Programming*.

OOP aimings at gathering several related data together, with functions describing how these data are processed and cooperate with each other, inner or outer. A collection of such things is called *object*, with data called *property* and function called *method*.

OOP, and other algorithm

Integrate single variable functions using plain Riemann or Darboux summation, should be somehow a simple thing. So we'll try to integrate double variable functions. This brings a problem: How can I define a double variable function? Here we introduce: *OOP*, *Object Oriented Programming*.

OOP aimings at gathering several related data together, with functions describing how these data are processed and cooperate with each other, inner or outer. A collection of such things is called *object*, with data called *property* and function called *method*.

Aside for the plain sum, there are indeed some better algotithms to do numerical integral. While, talk is cheap. Let's head for the code.

Nothing but Derivative

The title reveals it all. What we need is just extend the differential from one dimension to higher dimension.

Nothing but Derivative

The title reveals it all. What we need is just extend the differential from one dimension to higher dimension.

But it *is* the higher dimension matters. We are going to, again, use OOP to handle this problem.

Nothing but Derivative

The title reveals it all. What we need is just extend the differential from one dimension to higher dimension.

But it *is* the higher dimension matters. We are going to, again, use OOP to handle this problem.

We are going to implement the ∇, gradient and laplacian with Python, and utilise the feature of OOP.

Sum it up

What did we get by now?

- Numerical methods for solving differential equations, integrates, gradient and laplacian.
- Basic knowledge about programming.
- Some Python skill enabling one to do many things.

Resources

Here's a list of recommended resources.

- Math, Numerics, & Programming (for Mechanical Engineers) introduces a lot of practical numerical methods together with some programming implementation.
- My Github repository about Python developing environment set up with VS Code and Jupyter notebook.
- Runoob provides good introduction and basic usage of Python.
- Python documentation set for who interested in Python with details.
- *Python Crash Course* is a good Python book for beginers. Its resources collection is in Github .
- If you meet any question with any package/module, please refer to its documentation site.