Computer Practical: Tools of the Trade

Paul Connolly, September 2023

1 Overview

In this computer practical, you will investigate the solution of a ordinary differential equations using approximate methods on a computer. The computer programs are already written as Python scripts / files and you will edit them and run them under your log-in on a server computer. During the practical session you will open up and edit a Python script and try to understand various parts of the code. We do not need to understand the full details, but will use the practical to aid our understanding of how numerical models are put together. Throughout the practical you will be looking at Python scripts, editing the scripts and running them from the command line through an SSH terminal.

2 Code formulation

The code you will be running today is written in Python, which is a *interpreted language*. This means the computer needs to spend time translating each line of the code into its machine language before it can execute it. Therefore code written in Python can be slow. However, because computers are extremely fast, the computations will be done fairly quick.

Figure 1 illustrates the main idea behind the model simulations. We start with our *ordinary differential equation* to solve. An example here is the equation for radioactive decay:

$$\frac{dN}{dt} = -\lambda N \tag{1}$$

where N is the number of radioactive atoms, t is time, and λ is a constant (s^{-1}). This equation is applicable for all times, t. We then assume that our time variable, t, is discretized into m equally spaced values, $n \times \Delta t$ where n is a whole number, and we calculate the gradient using the *forward euler approximation*:

$$\frac{N_{n+1} - N_n}{\Delta t} = -\lambda N_n \tag{2}$$

Equation 2 says the same as Equation 1, except the gradient on the left-hand side is calculated differently. The subscript, n, means that this applies at all values along the discretized time axis. Now, if we know the value of N at the start, N_0, we will be able to calculate the next value of N on the discretized time axis by rearranging Equation 2 for N_{n+1} as follows:

$$N_{n+1} = N_n - \Delta t \times \lambda N_n \tag{3}$$

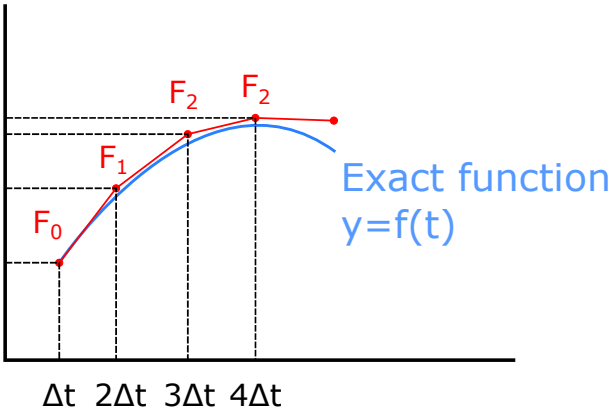

Figure 1: Schematic of the way we will be solving our equations using the *Forward Euler Method*. We calculate the gradient using the governing equation and estimate this gradient using a *finite-difference approximation*. This allows us to advance our solution forward in the x-coordinate.

if the subscript $n = 0$ then we can see how we can advance our value of N_0 to the next value along the discretized time axis all the way to the m^{th} point:

$$\begin{aligned} N_1 &= N_0 - \Delta t \times \lambda N_0 \\ N_2 &= N_1 - \Delta t \times \lambda N_1 \\ N_3 &= N_2 - \Delta t \times \lambda N_2 \\ \vdots & \quad \vdots \\ N_m &= N_{m-1} - \Delta t \times \lambda N_{m-1} \end{aligned}$$

the above, using the *forward Euler method*, is just one kind of *approximate method* or *approximate solution*. Many different methods exist. In general the results will differ to the actual solution, and we will use this practical to investigate.

3 Downloading the code

Log into the server computer with a SSH. Also, in a separate terminal or CMD window, log into the server computer with SFTP. Space the windows out on your screen so that you can easily move between them.
If you have not done this already, download the code you will be using today from GitHub by typing the following into an SSH window:

```
git clone https://github.com/EnvModelling/approximate-methods-practical
```

this should download the code to your working directory.
In order to access the files you will need to change the working directory to where the files are. Type the following in the SSH window:
`cd approx`
followed by the `tab` key and the command should auto-complete. Then press the `enter` key.

Finally, type `ls` followed by the `enter` key. The screen should list the files in this directory.

4 Viewing and Editing the code

We will use the `nano` text editor to view the files, although this time we will pass an extra option to show *line numbers*. For example, type:

```
nano -l approximate_methods01.py
```

the `-l` means to show line numbers. We will take a moment to look at what the code is doing. The code can be edited in the text editor and saved by typing `Ctrl-X` at the same time and pressing `Y` to save the file.

Table 1: Parameters we will change in the Python scripts.

Variable	Default for `approximate_methods01.py`	Default for `approximate_methods02.py`	Description
`dt`	1s	1s	The time-step for the analytical calculations.
`dtsolve`	10s	0.5	The time-step for the approximate method calculations.
`method1`	1	1	a flag that changes the way the code behaves.

5 Running the code

To run the model all that is required is to:

1. Edit the Python file you want to run.
2. Type `python3 <filename>` followed by `enter` where `<filename>` is the name of the file you want to run. For example:
   `python3 approximate_methods01.py`
   followed by the `enter` key
3. You will usually need to wait a short while until the code finishes running.

After running the code it will generate some an output file in a temporary file location on the server computer. The file location will be in `/tmp/<username>/`, where `<username>` is your actual username on the server computer.

6 Getting the output file

After your code has run, see what the output file is called by typing the following at the SSH command line.

`ls /tmp/<username>`

where `<username>` is your actual username. Once you know the name of the file go to the SFTP window and type:

`get /tmp/<username>/<filename>`

where `<filename>` is the name of your file. This will download the file to your computer where you will be able to open it up and view it.

I STRONGLY SUGGEST YOU OPEN UP A WORD OR POWERPOINT DOCUMENT AND INSERT THE FIGURES AND MAKE SOME NOTES AS YOU GO. THIS GOES FOR ALL PRACTICALS WHERE WE WILL BE RUNNING MODELS. YOU COULD BE ASKED QUESTIONS ABOUT THEM IN THE ASSESSMENT.

7 Experiments

7.1 Changing the time-steps in the Forward Euler Method

Run the `approximate_methods01.py` script in SSH and save the output file to your computer, using SFTP. Now, using the `nano -l` command edit the file `approximate_methods01.py` on line 23 so that the value of `dt=1000`. Run the model and get the output file using SFTP.

Change the value of `dt` back to the default `dt=1`
Repeat the simulations for:

- `dtsolve = 1000`
- `dtsolve = 10000`

Do not forget to save the images each time.

Question 1: How does increasing the value of `dt` change the analytical solution? Why?

Question 2: How does increasing the value of `dtsolve` change the Forward Euler solution? Why?

7.2 Second order ODE

In this simulation you will use `approximate_methods02.py`, which solves a 2nd order ODE. Run the simulation and save the file to your note books.
Repeat this simulation for `dtsolve=0.001`, and download / save the output.

Question 1: What do you notice about reducing the time-step for the approximate methods in these calculations?

Question 2: Generally, what happens to the approximate method solution when you reduce the time-step. Why do you think this is?

7.3 Partial Differential Equation

We have not gone into the detail for how we solve partial differential equations, but there is an example script to run here, which solves the transport equation (we will cover this later on). For now, lets just run the script and see what we get. It generates a different kind of image file called an animated 'gif', which allows us to visualise how the solution changes with time.

Run the `approximate_methods03.py` script and download the output file to your computer. If you open it in a web browser you will be able to see the animation. The code calculates the transport /movement of a triangular plume of pollution along the x-axis.

Discussion: What do you notice about the approximate method (known as the upwind scheme) vs the analytical method.

Bonus: Change the value of `dt=6` and run the simulation again. The solution blows up! This is because the method has become unstable. We will learn about a value of the 'CFL' number $CFL = v\frac{\Delta t}{\Delta x}$ where Δx is the discretization in space. If this number is greater than 1 some approximate methods that solve the transport equation become unstable.