Project and Report (2)

December 2023

The second "Project and Report" will ask you to create and run a model and then investigate the results of running this model. To achieve this, you should use the skills that you have developed so far in the Intensive Introductory Scientific Computing with Data Science unit. The model can be written in Python or C++, but the structure and efficiency of the model should be taken into account. You are expected to consider and justify your approach based on the material covered within the unit. Details of the scientific problem that you will investigate are given below.

The project should consist of a code component and a written report. You will have until **Wednesday, 24th January at 10 am** to complete this project and submit your work. This project will be worth 25 % of your total unit mark, weighted as 15 % on the code and 10 % on the report. This assessment will consider the code you write, including any documentation, code clarity, and programming style, in addition to the scientific discussion in the written report. While the report's subject focuses on a particular area of science, **the assessment of this work covers Scientific Computing and Data Science**; therefore, directions for background discussion are given in the details below.

While the model can be written in either Python or C++, it is expected that the analysis of the output of your model will be performed in Python. The code component should include all of the code used to run your model and perform your analysis, including any plots produced (either within a Jupyter Notebook or referenced as external files). The code must be self-contained, and it should be clear which parts of the analysis are performed at each stage. The code should be well commented and additional details can be added using separate Markdown cells (if using a Jupyter Notebook). **The code should be able to run as a whole independently (from top to bottom) or compiled and run, as relevant**.

For the report component, you should write this using the word processor of your choice and submit this alongside the code as a Word document (.doc or .docx) or as a PDF (.pdf). The report should include background details of any scientific context, a description of your model and analysis, discussion, and conclusions. You should also include a short abstract summarising your findings and details of any references used. See Section 2 for these details as bullet points. The report itself does not need to include explicit details of the code used. However, the overall approach and any relevant outputs should be related to the submitted code document. The separate code document can be referred to if needed.

When completed, use the "Project and Report 2" Blackboard submission point to upload your files (on the "Assessment, submission and feedback" course content area for the "Intensive Introductory Scientific Computing with Data Science" unit).

You will submit multiple files for this project (at least one code file and one report). All code files should be submitted in a runnable Python or compilable C++ format, e.g. a Jupyter

notebook (".ipynb"), Python file (".py"), or C++ source file (".cpp"). Other formats may not be accepted (such as ".html" and ".pdf" documents). **Please check the format for all files before submitting**.

All submissions for this assessment must have been created by the submitter and should not be created or copied using alternative resources (including copying from your peers or using large language models (LLM) tools to generate code or text). Please also see the University policy on plagiarism.

1 Orbits

You have been asked to assemble a model that can be used to simulate the movement of multiple celestial bodies under the forces of gravity. The main test case will focus on the Solar System by looking at the collective movement of the Sun, the Earth and the Moon. This is known as an N-body simulation where we look at each body/object, resolve the total force acting on this and then move each body/object. For this particular assessment, we will only ask for up to 3 bodies to be considered.

1.1 Methodology

During this unit, we covered several ways to solve these problems. This project can be approached using any of those tools (including `scipy` methods, using functional programming), but if this is completed using manual iterative steps, use the Verlet 'leap-frog' scheme (described below) rather than the Forward Euler method. The Verlet 'leap-frog' scheme is applied similarly to Forward Euler but will produce better and more stable results for this model.

1.1.1 2-dimensional

We want to consider our objects' movement in two dimensions: x and y (i and j are the basis vectors for these directions). This means we need to consider these two components for our force, acceleration, velocity, and position for each of our bodies.

The **unit vector** is a vector of length 1 that describes the direction of a vector. To calculate the unit position vector between bodies k and l, $\hat{\mathbf{r}}_{\mathbf{kl}}$, the following equation is used.

$$\hat{\mathbf{r}}_{kl} = \frac{\mathbf{r}_{kl}}{|\mathbf{r}_{kl}|}, \tag{1}$$

where, $|\mathbf{r_{kl}}|$ is the magnitude of the vector $\mathbf{r_{kl}}$.

Alternatively, the x and y dimensions are independent, so it is possible to compute these as a pair of scalars.

1.1.2 Resolving forces and acceleration

The force on each body from another body is given by:

$$\mathbf{F}_{kl}(t) = G\,\frac{m_k m_l}{|\mathbf{r}_{kl}(t)|^2}\,\hat{\mathbf{r}}_{kl}(t) \tag{2}$$

where,

- $\mathbf{F}_{kl}(t)$ is the force vector, at time t, between object k and l. This is written as $\mathbf{F}_{kl}(t)$ (with bold font) because this is a vector, where we need to consider the x and y components.
- m_k and m_l are the masses of the two objects in kg
- $\mathbf{r}_{kl}(t)$ is the position vector, at time t, between the two bodies (where $\mathbf{r}_{kl}(t) = -\mathbf{r}_{lk}(t)$) in m.

- G is the gravitational constant $6.67 \times 10^{-11}\,\mathrm{N\,m^2\,kg^{-2}}$.

Note that using the unit vector in Eqn. 2 imposes the direction of the force vector.

For each of the bodies, we need to resolve the total gravitational forces from all of the other bodies,

$$\mathbf{F}_k(t) = \sum_{l \neq k} \mathbf{F}_{kl}(t)\hat{\mathbf{r}}_{kl}(t). \tag{3}$$

For example, if there were two bodies, this would resolve to,

$$\begin{aligned} \mathbf{F}_1(t) = \mathbf{F}_{12}(t) &= G\frac{m_1 m_2}{|\mathbf{r}_{12}(t)|^2}\hat{\mathbf{r}}_{12}(t), \\ \mathbf{F}_2(t) = \mathbf{F}_{21}(t) &= G\frac{m_2 m_1}{|\mathbf{r}_{21}(t)|^2}\hat{\mathbf{r}}_{21}(t). \end{aligned} \tag{4}$$

Since $|\mathbf{r}_{12}(t)|$ and $|\mathbf{r}_{21}(t)|$ are the same (they are both the distance between bodies 1 and 2), the only difference between $\mathbf{F}_1(t)$ and $\mathbf{F}_2(t)$ is that they act in opposite directions, i.e., $\mathbf{F}_1(t) = -\mathbf{F}_2(t)$.

In the three-body case, this would become,

$$\begin{aligned} \mathbf{F}_1(t) = \mathbf{F}_{12}(t) + \mathbf{F}_{13}(t) &= G\frac{m_1 m_2}{|\mathbf{r}_{12}(t)|^2}\hat{\mathbf{r}}_{12}(t) + G\frac{m_1 m_3}{|\mathbf{r}_{13}(t)|^2}\hat{\mathbf{r}}_{13}(t), \\ \mathbf{F}_2(t) = \mathbf{F}_{21}(t) + \mathbf{F}_{23}(t) &= G\frac{m_2 m_1}{|\mathbf{r}_{21}(t)|^2}\hat{\mathbf{r}}_{21}(t) + G\frac{m_2 m_3}{|\mathbf{r}_{23}(t)|^2}\hat{\mathbf{r}}_{23}(t), \end{aligned} \tag{5}$$

and similarly for $\mathbf{F}_3(t)$.

Once the force vectors are found for each of the bodies, these can be used to calculate the acceleration vectors,

$$\mathbf{a}_k(t) = \frac{\mathbf{F}_k(t)}{m_k}. \tag{6}$$

This could alternatively be written without the vector notation, treating the x and y components individually,

$$\begin{aligned} a_{k,x}(t) &= \frac{F_{k,x}(t)}{m_k}, \\ a_{k,y}(t) &= \frac{F_{k,y}(t)}{m_k}. \end{aligned} \tag{7}$$

1.1.3 Leap-frog scheme

If completing this assignment using a manual iterative approach (i.e. not `scipy` or `solve_ivp`), the acceleration should be used to update the velocity and position of each body using the Verlet 'leap-frog' scheme. This scheme starts with calculating the force and acceleration on each body at $t = 0$, based on some initial position. Using this acceleration value and an initial starting velocity, the position at $0 + \delta t$ is found with

$$\mathbf{r}_k(0 + \delta t) = \mathbf{r}_k(0) + \mathbf{v}_k(0)\delta t + \frac{1}{2}\mathbf{a}_k(0)\delta t^2, \tag{8}$$

where, δt is the time step size. The time is then updated to $0+\delta t$, and the force and acceleration are computed. Using this new acceleration value, the new velocity can be found with,

$$\mathbf{v}_k(0+\delta t) = \mathbf{v}_k(0) + \frac{\delta t}{2}[\mathbf{a}_k(0) + \mathbf{a}_k(0+\delta t)]. \tag{9}$$

These equations can be written generally as,

$$\begin{aligned} \mathbf{r}_k(t+\delta t) &= \mathbf{r}_k(t) + \mathbf{v}_k(t)\delta t + \frac{1}{2}\mathbf{a}_k(t)\delta t^2 \\ t &\leftarrow t + \delta t \\ \mathbf{v}_k(t) &= \mathbf{v}_k(t-\delta t) + \frac{\delta t}{2}[\mathbf{a}_k(t) + \mathbf{a}_k(t-\delta t)], \end{aligned} \tag{10}$$

where, the second line indicates the updating of the time value.

1.2 Scientific Background

The modelling of planetary motion is a type of N-body modelling problem. **Discuss in your introduction some other applications of N-body modelling in the sciences and engineering**.

The orbit of the planets (including the Earth) are not perfectly circular; instead, they take an elliptical path. A common parameter that is used to describe the orbit of a planet is the eccentricity of the ellipse, e,

$$e = \frac{r_a - r_p}{r_a + r_p}, \tag{11}$$

where, r_a is the radius at apoapsis (the farthest distance between the planet and the Sun), and r_p is the radius at periapsis (the shortest distance between the planet and the Sun).

For every orbit the Earth takes around the Sun, the Moon orbits the Earth approximately 12 times, leading to the 12 months we have in the calendar. The orbital period of the Moon can be calculated by finding the difference between the Sun to Moon distance and the Earth to Moon distance and calculating the distance between two of the peaks (this may be achieved with the `scipy.signal.find_peaks` function). A plot of the difference between the Sun to Moon distance and the Earth to Moon distance is shown for a half-year dataset in Figure 1.

1.3 Technical Details

Here, we outline some technical details that are important for the completion of the assessment.

1.3.1 Animation Script

You will find a script entitled `animate_orbits.py` in the assessment information on Blackboard Learn. This script will produce an animation of the orbits of the Earth around the Sun and the Moon around the Earth, as shown in Figure 2. Note that this is representative of your simulation output and will include the entire time range, but the animate script **will not plot every time point** as this would be very slow to run. To control the number of frames, there is an optional command line argument that is discussed below.

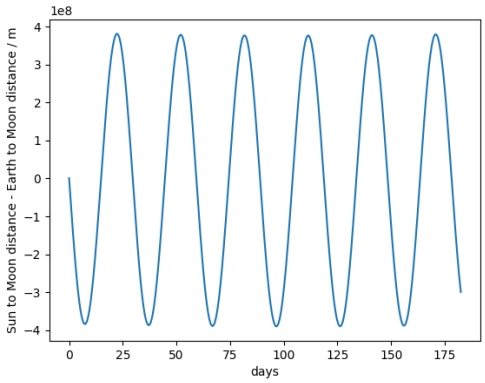

Figure 1: The difference between the Sun to Earth and Earth to Moon distances over half a year.

This script will produce a gif image showing the orbits, where the Sun is marked with an orange dot, the Earth in green and the Moon in red (where the path of the orbits are shown with solid lines). There are two panels: the left-hand panel shows the Earth's orbit around the Sun, and the right-hand side zooms in on the Earth and the Moon. The script should be stored in the same directory as the data file and can be run using Python on the command line (terminal).

This must be run from within your Anaconda installation to access the right Python packages (pandas, matplotlib, etc.). Using your Anaconda installation, open up a Terminal in JupyterLab (*NOT WSL/Ubuntu*).

Within the appropriate terminal, the animate_orbits.py code can be run with:

```
python animate_orbits.py --filename file_name.csv --frame n_frames
```

where `file_name.csv` should be replaced with the appropriate data file name (see below) and `n_frames` can be replaced with the number of frames (note that the latter will default to 100 if no value is given).

Note: you may need to use the `python3` *command rather than* `python` *depending on your installation.*

The data must be stored in the correct format for this script: a comma-separated values file with six columns. The first line of the file must read:

```
x1,y1,x2,y2,x3,y3.
```

This is followed by a line for each time point (in order), where the data are in the following columns:

- `x1` is the x-position of the Sun
- `y1` is the y-position of the Sun
- `x2` is the x-position of the Earth
- `y2` is the y-position of the Earth

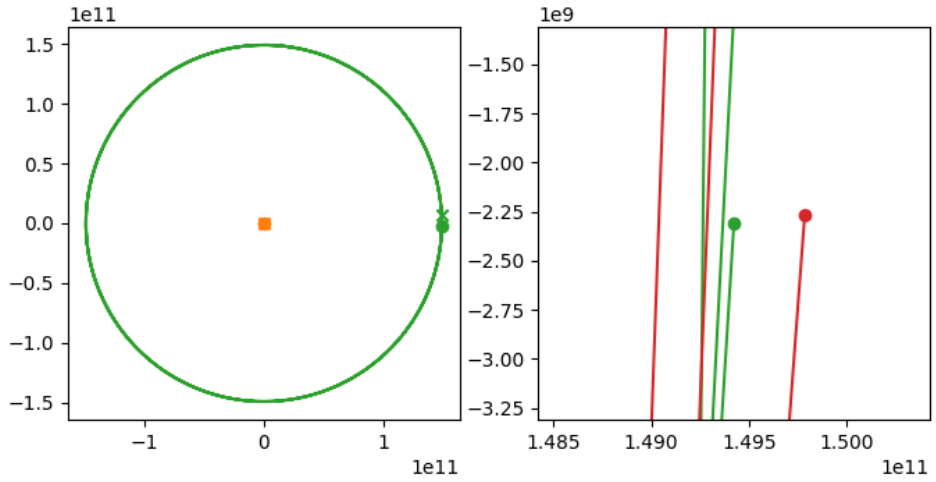

Figure 2: An example of a frame that is output by the `animate_orbits.py` script.

- `x3` is the x-position of the Moon
- `y3` is the y-position of the Moon

An example of such a file containing the x- and y-positions for only the Sun and Venus is available on the assessment information page of Blackboard Learn as `orbit_venus.csv`.

1.3.2 Writing to a File

Python: The correct data structure in Python can be written by creating a pandas DataFrame where the positions are stored in columns with the column headings defined above and using the `to_csv` method. When using this function, the `index=False` option must be specified to ensure only the x and y column details are written.

C++: Writing to a file in C++ requires a little more work. The code below shows, generally, how a file (containing, in this example, only the x-positions of three bodies) may be written.

```
#include <iostream>
#include <fstream>
#include <vector>

int main() {
    // Create and open a text file
    std::ofstream file_out("filename.txt");

    // Create a vector to write
    std::vector<double> a_vector = {1.4123, 3.1424};
    std::vector<double> b_vector = {7.3244, 5.3242};
```

```
    std::vector<double> c_vector = {3.7564, 2.8712};

    // Write to the file
    file_out << "x1" << "," << "x2" << ","
             << "x3" << std::endl;
    file_out << a_vector[0] << "," << b_vector[0]
             << "," << c_vector[0] << std::endl;
    file_out << a_vector[1] << "," << b_vector[1]
             << "," << c_vector[1] << std::endl;

    // Close the file
    file_out.close();
}
```

The above file writing code should be adapted appropriately for use in this assessment.

assessment: Construct a model for the three-body simulation of the Sun, the Earth and the Moon using the following starting conditions:

- $r_{\text{Sun}}(0) = (0\,\text{m},\ 0\,\text{m})$
- $r_{\text{Earth}}(0) = (1.495 \times 10^{11}\,\text{m},\ 0\,\text{m})$
- $r_{\text{Moon}}(0) = (1.495 \times 10^{11}\,\text{m},\ 3.84 \times 10^{8}\,\text{m})$
- $v_{\text{Sun}}(0) = (0\,\text{m},\ 0\,\text{m})$
- $v_{\text{Earth}}(0) = (0\,\text{m},\ 2.978 \times 10^{4}\,\text{m}\,\text{s}^{-1})$
- $v_{\text{Moon}}(0) = (-1.022 \times 10^{3}\,\text{m}\,\text{s}^{-1},\ 2.978 \times 10^{4}\,\text{m}\,\text{s}^{-1})$
- $m_{\text{Sun}} = 1.988\,47 \times 10^{30}\,\text{kg}$
- $m_{\text{Earth}} = 5.9772 \times 10^{24}\,\text{kg}$
- $m_{\text{Moon}} = 7.3476 \times 10^{22}\,\text{kg}$

Run the model for three years, with a timestep of one hour (note that to match the units of the starting conditions, the units of the timestep must be seconds). Check that your model produces a suitable orbit using the `animate_orbits.py` script and, using the output data file, calculate Earth's orbital eccentricity around the Sun and the period of the Moon.

Investigate the effect of changing the initial parameters on the stability of the system, such as:

- the impact of changing the mass of the Sun between 0.7 and 1.3 solar masses on the Earth's orbital eccentricity,
- the impact of changing the mass of the Moon between 1 and 100 lunar masses on the approximate orbital period of the Moon around the Earth.

1.4 Optional: Beyond the Brief

Within the separate mark descriptor document, you will see that completing the outline above to a good standard will allow you to achieve a very good mark. To start to move beyond this and achieve the very highest marks (excellent work), as well as fully completing the outline, your submission would need to offer additional quantification or comparison of the data (described as "beyond the brief"). Consider other analyses that can be performed from the collected data or how this model can be used to investigate exoplanetary systems.

2 Report

The report should be a substantial piece of work, so you should aim for your report to be approximately 1000 to 2000 words or 3 to 5 sides of A4 paper (including plots). Note that this is not a hard limit for the report; where possible, you should provide adequate detail and be concise and directed.

The report itself should be split into the following sections:

- Abstract: Short overview and summary of the key results of your analysis.
- Introduction: Overall formulation of the questions being asked and introduction of any necessary background.
- Analysis and Discussion: Details of the approach and analysis performed for the different parts of the investigation. This section should include plots or summary tables as appropriate.
- Conclusions: Any overall conclusions that can be drawn from your analysis.
- References: Full details of references used in the construction of this report (using the Royal Society of Chemistry citation format, see edu.rsc.org/resources/how-to-reference-using-the-rsc-style/1664.article for details).