PHAS0030: Computational Physics
Mini-project briefing: complex orbital dynamics

David Bowler

February 17, 2023

The precise solution of the orbital motion of bodies under the influence of gravity is an important and challenging problem. This project will consider realistic problems (models of the solar system) and idealised problems (complex orbits for bodies of the same mass), with the stability both of different physical situations and of different propagation methods being considered. After a guided introduction, you will have freedom to choose a research direction, and will apply the computational physics skills you have learned during the course to investigate these important problems.

Contents

1 Objectives

The objectives of this project are to:

- Investigate simple problems with two bodies moving under the influence of gravity (both with equal masses and with different masses)
- Extend this simulation to systems of several bodies (idealised and/or realistic) and perform independent research on this problem

- Explore the stability and accuracy of different solution methods for the differential equations, and the effect of initial conditions
- Gain experience in performing independent research
- Demonstrate your skills in using computational physics for a practical problem
- Show that you can write a formal report documenting the results of your project

2 *Introduction*

The gravitational force on a mass, m_1, at a point $\mathbf{r}_1$ due to another mass, m_2, located at a point $\mathbf{r}_2$, is given by:

$$\mathbf{F}_{12} = \frac{Gm_1m_2}{|\ \mathbf{r}_{12}\ |^3}\mathbf{r}_{12} \tag{1}$$

where $\mathbf{r}_{12} = \mathbf{r}_2 - \mathbf{r_1}$. From this simple force, we can find complex behaviours that are still being researched today (as seen in the references). Note that analytic solutions cannot be found for more than two bodies: all solutions to more complex problems must be numerical.

Systems of the kind to be studied in this project should conserve energy, and monitoring this is a key way to characterise the stability of a numerical method. The gravitational potential energy for a group of bodies with masses m_i is given by:

$$U = \sum_i \sum_{j>i} -\frac{Gm_im_j}{|\ \mathbf{r}_{ij}\ |} \tag{2}$$

where the sum over j is chosen to avoid double counting of terms. The kinetic energy is the usual form, $\sum_i \frac{1}{2}m_iv_i^2$. Angular momentum should also be conserved, and is easily calculated for simple systems.

3 *Preliminary investigation*

This section includes topics that you *must* address in your final report, while the later sections offer suggestions of what you might want to consider, but do not require anything specific. You should aim to do most of the work in this section before reading week, so that you can get help on any problems that you are having (note that the Week 4 assignment forms the first part of this work). The basic techniques you will implement here will form the basis of all the work in your mini-project, so ensure that you are confident in what you are doing. We will work in a set of units (which I will not define explicitly) which have G=1, for simplicity. By the end of this preliminary work, you should have successfully implemented orbital dynamics for two and three body systems, and characterised the stability of two different techniques.

3.1 *Integrators: background information*

The simplest integrator that we will use is Euler's method, which is described in full detail in the Week 4 notes. If we have a function $f(t)$ and its time derivative, df/dt, then we can advance it in time by a small amount, dt, using the following approach:

$$f(t+dt) = f(t) + dt\frac{df}{dt} \tag{3}$$

This way, we can calculate the value of the function from a known starting point, $f(t_0)$, at a set of times $f(t_1), f(t_2), \ldots$ where $t_1 = t_0 + dt, t_2 = t_1 + dt = t_0 + 2dt, \ldots$ etc. More specifically, we can calculate the position $\mathbf{r}_i$ and velocity $\mathbf{v}_i$ for a body i with mass m_i using this method with the equations $d\mathbf{r}_i/dt = \mathbf{v}_i$, $d\mathbf{v}_i/dt = \mathbf{F}_i/m_i$, where we have calculated the acceleration from the force acting on the body.

A simple implementation of the Euler method in a 1D system might look like this (assuming the existence of a Python function `dfdt(f,t)`):

```
# Parameters
N = 1001
dt = 0.01
# Storage
t = np.linspace(0,N*dt,N+1)
f = np.zeros(N+1)
# Initialise and propagate
f[0] = 0.0
for i in range(N):
    f[i+1] = f[i] + dt*dfdt(f[i],t[i])
```

However, as we have seen in Week 4, the Euler method is inherently unstable, and requires a small timestep for accuracy; even with a small timestep, instabilities often occur at long times. A more stable and accurate method, which is often used when modelling systems of particles, is the velocity Verlet method, which will be formally introduced in Week 7. It conserves energy, and is time reversible (it is one of a class of methods known as symplectic methods). The equations are a little more complicated than for Euler's method, and require a calculation of the force at time $t + dt$ *before* the velocity at that step can be found:

$$\mathbf{r}_i(t+dt) = \mathbf{r}_i(t) + dt\mathbf{v}_i(t) + \frac{1}{2}dt^2\frac{\mathbf{F}_i(t)}{m_i} \tag{4}$$

$$\mathbf{v}_i(t+dt) = \mathbf{v}_i(t) + \frac{1}{2}dt\left(\frac{\mathbf{F}_i(t) + \mathbf{F}_i(t+dt)}{m_i}\right) \tag{5}$$

A simple implementation of the velocity Verlet method for a 2D system with one particle, assuming a Python force function `force(r)`, might be:

```
# Parameters
N=1001
dt = 0.01
m = 0.0001
# Storage
t = np.linspace(0,N*dt,N+1)
r = np.zeros((N+1,2)) # N+1 timesteps, 2 components
v = np.zeros((N+1,2))
# Initial conditions
r[0] = np.array((1.0,0.0))
v[0] = np.array((0.0,1.0))
# Propagate
for i in range(N):
    # Force at step i
    fi = force(r[i])
    # Position at step i+1
    r[i+1] = r[i] + dt*v[i] + 0.5*dt*dt*fi/m
    # Force at step i+1, using new position
    fi_plus_1 = force(r[i+1])
    # Velocity at step i+1
    v[i+1] = v[i] + 0.5*dt*(fi + fi_plus_1)/m
```

Note that the choices for number of steps, timestep, mass, and initial conditions in the two examples above are entirely arbitrary, and you should be sure to use the correct values for your simulation.

The velocity Verlet method is accurate to second order in the timestep (errors are proportional to dt^2), while the Euler method is only accurate to first order in the timestep. There are methods which are higher order in timestep, such as the Runge-Kutta methods, but these often are neither time-reversible nor energy conserving. We have discussed the most common of these, the fourth order RK4 method, in Week 4, where we created a function to implement this method; it is also possible to use a function from SciPy, `integrate.solve_ivp`, though this requires careful setting up, and will be discussed in detail below. But in orbital dynamics, energy conservation and time reversibility is often more important than the higher accuracy (and hence longer timesteps) that come from higher order methods.

3.2 *A first test: two bodies, different masses*

This task, and the next, form the assessment for Week 4 of the course, but you can reuse that work as part of the mini-project. Here, we will consider a system like a moon orbiting a planet, or a planet orbiting a star. You should start with a large mass ratio (at least 1,000:1), setting $G = 1$ and $m_1 = 1$ for the large body, and use Euler's method (even though we know that it will not be a good approach). We will assume that the large body does not move, so put it at the origin, and do not update its position; put the small mass

at $r_2 = 1$. You should be able to work out the initial velocity for the small mass to move in a circular orbit around the large one using simple mechanics ($m_2 v_2^2/r_2 = G m_1 m_2/r_{12}^2$ but you should explain where this comes from).

You should then propagate the system forward in time for at least two orbits, recording the position and velocity at each timestep. Make appropriate plots of the trajectory. You can experiment with the timestep to find one that allows reasonable performance. You should also calculate the total energy at each step (kinetic and potential) either during the propagation or afterwards, and find the magnitude of the angular momentum at each step. Plot both of these, and check how well they are conserved, particularly with timestep size. Discuss your findings.

3.3 Two bodies, similar masses

This task is also part of the Week 4 assessment, but can be re-used. You will consider two bodies with similar masses (no more than a 1:10 ratio between them) moving about their centre of mass, which is formally defined as:

$$\mathbf{r}_{\mathrm{CoM}} = \frac{m_1 \mathbf{r}_1 + m_2 \mathbf{r}_2}{m_1 + m_2} \tag{6}$$

The equations are much simpler if you put the centre of mass at the origin ($\mathbf{r}_{\mathrm{CoM}} = 0$). Setting the distance between the bodies $r_{12} = 1$ and starting the bodies on the x-axis initially, it is easy to find the initial positions, while simple classical mechanics gives appropriate initial velocities for circular orbits around the centre of mass (for body 1, you need to solve $m_1 v_1^2/r_1 = G m_1 m_2/r_{12}^2$). Propagate the system forward in time using the velocity Verlet method, ensuring that you cover several periods of the orbit, and plot the orbits appropriately. Calculate the energy and angular momentum as before (though in this case you will need to add the energy and angular momentum for both bodies), and discuss their conservation. How do the Euler and velocity Verlet methods compare? You might like to examine the effect of the timestep on the orbits and the conservation laws.

3.4 Three bodies

We now extend the initial part of the project by going beyond the initial week 4 assignment; this is still a required part of the project, however.

We will consider a three body problem, which is a more challenging problem to initialise and propagate. We will use a system with a star, a planet and a moon, with masses $m_1 = 1$, $m_2 = 3 \times 10^{-6}$ and $m_3 = 3.6 \times 10^{-8}$ (again with G=1). We will set $r_{12} = 1$ and $r_{23} = 0.0025$. You should start the bodies aligned along an axis, choose a suitable origin and calculate appropriate initial velocities. A simple approximation would be to set $v_1 = 0$, and find v_2 in

the usual way, ignoring the influence of m_3; you can then find the velocity of m_3 *relative* to m_2 by considering the orbit of m_3 about m_2 (ignoring the influence of m_1) and then adding v_2 to v_3 to get an initial value for v_3 (you should check that you understand how I have arrived at these approximations, and that you can explain them for yourself—they are all based on simple orbital mechanics).

A more accurate approximation, but one which is a little harder to work out, would be to consider the orbit of a combined mass $m_{23} = m_2 + m_3$ about m_1 (the combined m_{23} will be located at the centre of mass of m_2 and m_3, and m_1 and m_{23} will orbit the centre of mass of the entire system) and then consider the orbit of m_2 and m_3 about their centre of mass. Explain which choice you have made, and give the background maths.

Propagate your system forward for at least two orbits of m_2 about m_1 using velocity Verlet, and consider how to plot the dynamics of all three bodies. You should also carefully consider the appropriate conservation laws, and demonstrate how well they are obeyed.

3.5 A different integrator

The final task in the first part of the mini-project is to evolve one of the systems you have already examined in Sections 3.2—3.4 using a higher order integrator (e.g. the fourth order Runge-Kutta integrator). You should choose a system that you are confident in working with, and can use either the Runge-Kutta code from the exercises (please make sure that you reference this appropriately) or you can call the function `integrate.solve_ivp` after importing the appropriate SciPy module (`from scipy import integrate`). The main challenge with using the SciPy function that I found was writing the function to pass to the integrator (the right hand side); this just requires a little care with the array that you pass which will include vector positions and velocities for all bodies that you are considering. Remember also that you can pass an array of times at which you want the solution evaluated (which can help make good plots). You should choose a combination you are confident with (the first task in Section 3.2, with the version of Runge-Kutta you did in class should be relatively straightforward, while the other tasks or the SciPy integrator may be more challenging). You could examine how well energy and angular momentum are conserved with the integrator, and perhaps its stability with timestep.

Let's consider the arrays in a little more detail. The SciPy function requires a 1D input for `y0` (the initial conditions); if you choose to do one particle in 2 dimensions (the problem in Sec. 3.2 where we fix the central body) then you will need an array with four components (position and velocity in two dimensions each). For two bodies, both moving in two dimensions (the problem in Sec. 3.3) you will need an array with eight components (position and velocity in two dimensions for each body). You can pack these in

any order you choose (though it should be logical to help others read the code). Your right-hand side function will need to unpack this input vector, calculate the appropriate derivatives, and pack them into an output vector of the same shape, with the same order. (To unpack, you can do something like `r1 = y[0:Ndim]` to put an Ndim-dimensional vector into `r1`, which can be passed to your force routine, and the reverse to create the output vector.) The final output from the SciPy function is ordered by component, and then timestep (so that if you call it with `result = integrate.solve_ivp(...)` then `result.y[0]` contains an array of the first component at all timesteps).

For your own routine (e.g. using the code from Week 4, which should be properly referenced if used) you are free to choose whatever array shape you find best. You could use a one dimensional array as SciPy does, or you could have an input array with shape `y0 = np.zeros((Npart,2,Ndim))` for `Npart` bodies (we have two components for position and velocity, each of which has `Ndim` dimensions). This can get a little cumbersome when adding a time dimension as well! But feel free to use whatever you find easiest.

4 *Progress check*

At this stage, you should now have written code which propagates the motion of two bodies over time, using both the Euler and the velocity Verlet methods (though note that you should not use the Euler method after this point—check that you understand why) as part of the Week 4 assessment. You should also have studied a simple three-body system, and a Runge-Kutta integrator for one of these early systems. (The Runge-Kutta integrator is the least important of these tasks, so if you are struggling to make it work, you would be better advised to work on the extensions.)

This basic approach to orbital dynamics should serve you for the rest of the project, so ensure that you are completely clear in what you are doing: it will become more complicated as you add more bodies to the simulation. You should also have some discussion of the different approaches and their accuracy, particularly in relation to conservation of quantities and effect of initial conditions. All this work, both coding and writing, can act as the basis for part of your final report, though of course it will need to be expanded and incorporated into the report.

5 *Extensions*

There are many ways in which the basic simulations given so far could be extended, but here we suggest two, and give extra references to help implement these. You can of course choose your own extension, but you will need to find references and data yourself. However you extend the project, remember to concentrate on the *physics* that you are studying, and how the computational im-

plementation enables insight into it. Your investigation should of course include tests of the implementation, and could touch on the reliability of the method, but physics is most important here.

5.1 Solar system dynamics

The orbital dynamics of the solar system are rich and varied, and provide a wealth of possible investigations. For instance, the eccentricity of the Moon's orbit around the Earth has a periodic variation over approximately 18 years called the Saros cycle (a fact known to the Babylonians, but less well known these days); you should be able to observe this in a simulation of the Sun-Earth-Moon system, using the correct orbital parameters. You might instead (or as well) choose to simulate more of the solar system (all planets as far as Mars or Jupiter, or beyond), or investigate the effect of the larger asteroids on the orbit of Mars (Ceres, which is officially a dwarf planet, and Vesta, for instance).

In order to successfully simulate the dynamics, you will need positions and velocities of the bodies you are simulating at a specific starting point. Fortunately, NASA publishes detailed tables of ephemerides (positions and velocities of bodies in the solar system over time) which you can use for this purpose. I used Table 5 in the Planetary and Lunar Ephemerides DE430 and DE431[1] (there is a more up-to-date reference[2] but that does not contain tables). You can also use the JPL Horizons system (`https://ssd.jpl.nasa.gov/horizons/app.html`) though this may require a little experimentation to understand. You can of course use a less accurate approach, using known orbital periods and distances of the Sun-Earth-Moon system, along with the average eccentricity, to estimate initial positions and velocities, adding the inclination of the Moon's orbit (5.1°) if you want; in the early stages I would advise doing this, starting without the inclination, to ensure that you are confident in the simulation. Once you are confident that it is working, you can add in the inclination, and turn to the exact positions and velocities if you wish.

[1] William M Folkner, James G Williams, Dale H Boggs, Ryan S Park, and Petr Kuchynka. The planetary and lunar ephemerides DE430 and DE431. *Interplanetary Network Progress Report*, 196(1):42–196, 2014

[2] Ryan S. Park, William M. Folkner, James G. Williams, and Dale H. Boggs. The JPL Planetary and Lunar Ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3):105, 2021

You will need to set up the appropriate constants and units (I would recommend working in SI units for simplicity, though whatever you choose, be sure to document it carefully). You should give careful thought to the timestep you will use, and the length of the simulation.

One target for the Sun-Earth-Moon system could be to reproduce the Saros cycle (seen in the variation of the eccentricity of the Moon's orbit). The eccentricity of an orbit can be measured by the Laplace-Runge-Lenz-Pauli vector:

$$\mathbf{L} = \frac{\mathbf{r}_R}{r_R} - \frac{v_R^2 \mathbf{r}_R - (\mathbf{r}_R \cdot \mathbf{v}_R)\mathbf{v}_R}{G(M_E + M_M)} \tag{7}$$

where the subscript R means the relative Earth-Moon position or velocity. You may find it easiest to see the periodic variation if you

plot the components of the eccentricity rather than the magnitude; you should also think about ways to analyse the eccentricity to find the frequencies (and hence the periods) of the variation (this potentially draws on material in Week 9 of the course, so might be best left to the end).

From here, the direction you take is up to you, though you should be cautious and ensure that you identify a problem that you can solve in a sensible time. You should also think carefully about the goal of your simulations, and try to identify a clear research question. This does not need to be very grand (for instance, identifying cycles in the Moon's eccentricity, and maybe their sensitivity to initial conditions, the integrator used or the time step) but should be scientifically meaningful.

5.2 N-body choreographies

Another very interesting area where you could choose to extend your investigation is that of orbits of multiple bodies with the same mass. The oldest of these problems, for three bodies, was solved by Euler and Lagrange; the Lagrange solution consists of the three bodies arranged in an equilateral triangle (these are known as the Lagrange points in general—the L2 Lagrange point which comes from the solution of a slightly different problem is well-known as the site of the James Webb telescope). But there are more complex orbits which can be found, which have been labelled choreographies, and they have been extended beyond three bodies as far as eight and beyond. The simplest example is for three bodies which all move in the same figure-of-eight pattern[3] (note that this paper is found here: `http://www.jstor.org/stable/2661357`). Patterns where all bodies follow the same path (though with different phases) are known as simple choreographies.

[3] Alain Chenciner and Richard Montgomery. A remarkable periodic solution of the three-body problem in the case of equal masses. *Annals of Mathematics*, 152(3):881–901, 2000

It is relatively easy to work out the correct initial velocity for the simple Lagrange problem with three bodies evenly spaced around a circle, maintaining circular orbits. Once you have found this, you could explore the stability and shape of orbits which do *not* match this initial velocity: what orbits do they take on, and when are they stable? It is also perfectly possible to extend this problem to N bodies evenly spaced around a circle: does dependence of orbits on initial conditions (particularly stability of the orbits) change with number of bodies?

If you want to investigate other simple choreographies, it is possible to find initial conditions for braids (shapes which extend the figure of eight to more bodies). There is data for 3-body, 4-body and 6-body braids[4], though it requires a little work to understand. Table 5 of that paper gives the parameters $\bar{x}$ and a for the four body braid; the definition of $E(x_1, \dot{x}_0, \dot{y}_1)$ (Eq 5.1) indicates how a and the components of $\bar{x}$ can be mapped onto the necessary initial conditions for all particles. Similarly, Eq. 6.1 and Table 7 give parameters for the six body braid. The parameters for the three body braid

[4] Tomasz Kapela and Piotr Zgliczyński. The existence of simple choreographies for the N-body problem—a computer-assisted proof. *Nonlinearity*, 16:1899, 2003

(simple figure of eight) are also given in the paper: $E(x_0)$ is defined just above Remark 3.4, and the components of the velocity for the first body (the only necessary conditions) are given just below Table 1; note that these parameters are given to many decimal places deliberately. Going beyond these problems is likely to be highly non-trivial; if you can successfully implement the 3-, 4- and 6-body braids, you could investigate their long term stability (in some cases this is barely one period), or the sensitivity to initial conditions or propagator. You should also check conservation of energy and angular momentum.

More recently, a large family of complex three body choreographies (i.e. where the bodies follow different paths) have been discovered[5], with a more detailed discussion of the searching procedure available[6] (Table 2 in this second paper gives a set of stable orbits similar to the figure eight braid—this might form an interesting basis for investigation). It is also possible to find orbits with non-zero angular momentum[7], with a similar searching approach (with footnote 13 giving initial conditions for an unusual three-body choreography). Note that these papers are quite advanced, and you should not spend a long time trying to understand the detailed mathematics—they are mainly provided for the orbits they describe.

These choreographies are very interesting, and it can be easy to be distracted creating appealing images of orbits. While meaningful representation of the orbit is important, you should always remember to keep one eye on the physics of the investigation: what are you learning? What are you testing?

[5] Milovan Šuvakov and V. Dmitrašinović. Three Classes of Newtonian Three-Body Planar Periodic Orbits. *Phys. Rev. Lett.*, 110:114301, 2013

[6] Milovan Šuvakov and V. Dmitrašinović. A guide to hunting periodic three-body orbits. *Am. J. Phys.*, 82(6):609–619, 2014

[7] Marija R. Janković, V. Dmitrašinović, and Milovan Šuvakov. A guide to hunting periodic three-body orbits with non-vanishing angular momentum. *Computer Physics Communications*, 250:107052, 2020

6 *Advice and suggestions*

Successful completion of a mini-project is challenging, and hopefully also enjoyable. This gives you a taste of independent research, and your final report should be a formal scientific report. I provide advice and suggestions for your work on the mini-project based on several years of supervision and marking.

- Build in complexity gradually: ensure that you understand what you have done in a given simulation before proceeding
- Keep back-ups: in particular, if you have a working implementation of a given problem which you are intending to change, keep a copy
- Don't be too ambitious: a well-documented, successful project will get better marks than an over-ambitious project that does not succeed
- Be sure to work steadily throughout term: you cannot do a project like this well in a short time
- Keep your log-book up-to-date

- Follow the principles of academic integrity: ensure that all work presented is your own, or is properly referenced

This year, owing to the class size, we cannot offer detailed individual or group supervisions as we have in the past. However, there will be a drop-in session during reading week (Monday February 13th, 2-5.30pm in Lab 3) where you can bring any work you have done on the first part of the project for help and discussion with demonstrators. After that, demonstrators will check on your progress during class, but will not be able to offer long, detailed sessions—instead, they will offer advice on how to address problems, and suggestions on how to develop the project where necessary.

7 *Submission*

Your final submission should consist of a word-processed report (as a PDF file) and one or more log books (which are generally Jupyter notebooks). The report should be no more than 20 pages, and should be in the form of a scientific report or paper.

You have freedom to choose how to write your report, but there are certain elements that should feature. You should provide an introduction to the topic, give an overview of the approach that you used (methods), present the results you found, and discuss their implications, and finish with a conclusion and a reference section.

Your log books should clearly document how your code and the project has developed, including how you developed and tested your code, discuss how you have overcome problems you have encountered, and show how your thoughts and ideas have developed; the work should all be dated clearly, to illustrate your progress through the project. Also note that you should be honest in the log books: seeing how you have overcome problems is an important part of the marking process.

References

[1] William M Folkner, James G Williams, Dale H Boggs, Ryan S Park, and Petr Kuchynka. The planetary and lunar ephemerides DE430 and DE431. *Interplanetary Network Progress Report*, 196(1):42–196, 2014.

[2] Ryan S. Park, William M. Folkner, James G. Williams, and Dale H. Boggs. The JPL Planetary and Lunar Ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3):105, 2021.

[3] Alain Chenciner and Richard Montgomery. A remarkable periodic solution of the three-body problem in the case of equal masses. *Annals of Mathematics*, 152(3):881–901, 2000.

[4] Tomasz Kapela and Piotr Zgliczyński. The existence of simple

choreographies for the N-body problem—a computer-assisted proof. *Nonlinearity*, 16:1899, 2003.

[5] Milovan Šuvakov and V. Dmitrašinović. Three Classes of Newtonian Three-Body Planar Periodic Orbits. *Phys. Rev. Lett.*, 110:114301, 2013.

[6] Milovan Šuvakov and V. Dmitrašinović. A guide to hunting periodic three-body orbits. *Am. J. Phys.*, 82(6):609–619, 2014.

[7] Marija R. Janković, V. Dmitrašinović, and Milovan Šuvakov. A guide to hunting periodic three-body orbits with non-vanishing angular momentum. *Computer Physics Communications*, 250:107052, 2020.