PH2150A Scientific Computing Skills Team Project
Computing planetary motion with the Runge–Kutta method

1 Introduction

In this project you will investigate the motion of a planet by solving the equations of motion numerically. In addition to using the usual $1/r^2$ law for the gravitational force, you will compute the effect of a correction due to general relativity.

Consider a planet of mass m moving under the force $\mathbf{F}$ due to the gravitational attraction of the sun. The equations of motion can be written in rectangular coordinates in the (x, y) plane as

$$\begin{aligned}
\dot{x} &= v_x && (1)\\
\dot{y} &= v_y && (2)\\
\dot{v}_x &= \frac{1}{m} F_x(x, y) && (3)\\
\dot{v}_y &= \frac{1}{m} F_y(x, y) \ . && (4)
\end{aligned}$$

This is a system of four coupled first order differential equations which we can solve numerically. The numerical methods are based on the idea that equations (1)–(4) can be used to relate $x(t)$, $y(t)$, $v_x(t)$ and $v_y(t)$ to their values at a slightly later time, $t + \Delta t$. By starting at given initial conditions and moving in small steps of Δt, the solution can be determined. Details on how to do this are given below.

Taking the sun (mass $M \gg m$) at the origin, the Newtonian gravitational force is given by

$$\begin{aligned}
F_x &= -\frac{mMG}{r^2} \cos\theta = -\frac{mMGx}{(x^2 + y^2)^{3/2}}\\
F_y &= -\frac{mMG}{r^2} \sin\theta = -\frac{mMGy}{(x^2 + y^2)^{3/2}} \ , && (5)
\end{aligned}$$

where the factors of $\cos\theta = x/\sqrt{x^2 + y^2}$ and $\sin\theta = y/\sqrt{x^2 + y^2}$ give the components in the x and y directions, respectively. Effects of general relativity can be included by means of a correction term for the magnitude of the force. The usual $1/r^2$ law becomes

$$F(r) = \frac{mMG}{r^2} + \frac{3}{2} \frac{mM^2G^2}{c^2 r^3} \qquad (6)$$

where c is the speed of light. The relativistic effects are only noticeable for the planets nearest to the sun, and they were first observed for Mercury. The effect of the additional term is to cause a precession of the perihelion of the orbit.

2 The assignment

Your assignment is to solve for the radius and angle of the orbit of Mercury about the sun as a function of time. In particular, you should:

- find out as much as you can about the solutions $x(t)$ and $y(t)$) (or equivalently $r(t)$ and $\theta(t)$) obtained with different numerical techniques. These should include the Euler method and two variations of the Runge–Kutta method (described below);
- display the solutions a plot of r versus t and also plot the planet's trajectory in space;
- determine the rate of precession of the perihelion of Mercury's orbit and to compare it with the measured value;
- investigate how the accuracy of the solution depends on the step size Δt for each of the methods used.

You will need to determine the initial conditions such that they correspond to the observed total energy and angular momentum of Mercury's orbit. You will need to look up various quantities such as Mercury's semimajor axis, period of rotation, mass, and the mass of the sun. For the purposes of testing the procedure, you can increase the magnitude of the relativistic correction so that it gives an easily visible effect.

3 The Runge–Kutta method

Our goal is to develop a numerical technique for solving systems of coupled first order differential equations. For now consider a single differential equation of the form

$$\dot{x} = f(t, x) \ . \tag{7}$$

Once we have seen how to solve this, generalising the technique to the case of four coupled equations will be straightforward. In order to solve (7), consider discrete steps in time of size Δt. The time after n steps is

$$t_n = t_0 + n\Delta t \ , \tag{8}$$

and the initial conditions are given as $x(t_0) = x_0$. Using the notation $x_i = x(t_i)$, we can approximate the solution at time t_{n+1} in terms of the solution at t_n by

$$x_{n+1} \approx x_n + \Delta t\, \dot{x}(t_n) = x_n + \Delta t\, f(t_n, x_n) \ . \tag{9}$$

By starting at $x(t_0) = x_0$ and repeating the rule (9) with a sufficiently small step size Δt, the solution $x(t)$ can be found. This is called the *Euler method.* It is not widely used in practice since far better methods (namely, the Runge–Kutta method) are available, but it contains the main idea. Since the approximation for x_{n+1} is based on a 1st order Taylor expansion about the point x_n, you can easily show that the error is proportional to $(\Delta t)^2$.

Now consider the four coupled differential equations (1) – (4) that describe the planetary motion. Suppose that initial values for x, y, v_x and v_y are given at time t_0. The positions and velocity components are then updated from step n to $n+1$ through the rule

$$
\begin{aligned}
x(t_{n+1}) &\approx x(t_n) + \Delta t\, \dot{x}(t_n) && (10)\\
y(t_{n+1}) &\approx y(t_n) + \Delta t\, \dot{y}(t_n) && (11)\\
v_x(t_{n+1}) &\approx v_x(t_n) + \Delta t\, \dot{v}_x(t_n) && (12)\\
v_y(t_{n+1}) &\approx v_y(t_n) + \Delta t\, \dot{v}_y(t_n)\ . && (13)
\end{aligned}
$$

To obtain, for example, $\dot{v}_x(t_n)$ for the right-hand side of (12), one evaluates the corresponding right-hand side from (3) using $x(t_n)$ and $y(t_n)$.

An improvement over equations (10)–(13) is provided by the Runge-Kutta method (pronunciation: Roong-eh-Kutta). Consider again the problem of a single differential equation of the form of equation (7). The problem with the Euler method was that the approximation for $x(t_{n+1})$ used the derivative $\dot{x}$ evaluated at the time t_n. But in the course of stepping from t_n to t_{n+1}, the derivative changes. A better guess would use the value of $\dot{x}$ averaged over the interval between t_n and t_{n+1}. As an approximation to this we can use the derivative at a point half-way between t_n and t_{n+1}, i.e. at $t_{n+\frac{1}{2}} = \frac{1}{2}(t_n + t_{n+1})$. Our rule is then

$$x_{n+1} = x_n + \Delta t\, f(t_n + \tfrac{1}{2}\Delta t, x(t_n + \tfrac{1}{2}\Delta t))\ . \tag{14}$$

The problem is that we don't have $x_{n+\frac{1}{2}} = x(t_n + \frac{1}{2}\Delta t)$. For this, however, we can use the approximation

$$x_{n+\frac{1}{2}} \approx x_n + \tfrac{1}{2}\Delta t\, f(t_n, x_n)\ . \tag{15}$$

That is, our method of finding x_{n+1} from x_n now consists of the following steps:

$$
\begin{aligned}
k_1 &= \Delta t\, f(t_n, x_n) && (16)\\
k_2 &= \Delta t\, f(t_n + \tfrac{1}{2}\Delta t, x_n + \tfrac{1}{2}k_1) && (17)\\
x_{n+1} &= x_n + k_2\ . && (18)
\end{aligned}
$$

Note that we still only needed the values of x at the point t_n to get x_{n+1}. You can show that the error from this method is proportional to $(\Delta t)^3$.

The most widely used version of the Runge–Kutta method uses the same basic idea, but a more sophisticated formula for the average derivative, based on the derivatives at the beginning, middle and end of the step. The algorithm is

$$
\begin{aligned}
k_1 &= \Delta t\, f(t_n, x_n) \\
k_2 &= \Delta t\, f(t_n + \tfrac{1}{2}\Delta t, x_n + \tfrac{1}{2}k_1) \\
k_3 &= \Delta t\, f(t_n + \tfrac{1}{2}\Delta t, x_n + \tfrac{1}{2}k_2) \\
k_4 &= \Delta t\, f(t_n + \Delta t, x_n + k_3) \\
x_{n+1} &= x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \; .
\end{aligned} \tag{19}
$$

One can show that the error in the solution with this method is proportional to $(\Delta t)^5$.

Consider now our four coupled differential equations for planetary motion (1) – (4). They are of the general form

$$
\begin{aligned}
\dot{x} &= f_1(t, x, y, v_x, v_y) && (20) \\
\dot{y} &= f_2(t, x, y, v_x, v_y) && (21) \\
\dot{v}_x &= f_3(t, x, y, v_x, v_y) && (22) \\
\dot{v}_y &= f_4(t, x, y, v_x, v_y) \; , && (23)
\end{aligned}
$$

where the functions $f_1, \ldots, f_4$ can be read off from the equations of motion (1) – (4). These equations can be written in a more compact notation as

$$
\dot{\mathbf{u}} = \mathbf{f}(t, \mathbf{u}) \; , \tag{24}
$$

where the four-component vectors $\mathbf{u}$ and $\mathbf{f}$ are defined as

$$
\begin{aligned}
\mathbf{u} &= (x, y, v_x, v_y) \; , && (25) \\
\mathbf{f} &= (f_1(t, \mathbf{u}), \ldots, f_4(t, \mathbf{u})) \; . && (26)
\end{aligned}
$$

As in the one-dimensional case, the solution can be found using the Runge–Kutta method once we are given initial conditions $\mathbf{u}(t_0) = \mathbf{u}_0$. The solution at step $n+1$ is obtained from that at step n by

$$
\begin{aligned}
\mathbf{k}_1 &= \Delta t\, \mathbf{f}(t_n, \mathbf{u}_n) \\
\mathbf{k}_2 &= \Delta t\, \mathbf{f}(t_n + \tfrac{1}{2}\Delta t, \mathbf{u}_n + \tfrac{1}{2}\mathbf{k}_1) \\
\mathbf{k}_3 &= \Delta t\, \mathbf{f}(t_n + \tfrac{1}{2}\Delta t, \mathbf{u}_n + \tfrac{1}{2}\mathbf{k}_2) \\
\mathbf{k}_4 &= \Delta t\, \mathbf{f}(t_n + \Delta t, \mathbf{u}_n + \mathbf{k}_3) \\
\mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} \; ,
\end{aligned} \tag{27}
$$

where there is a now a value of k_1, k_2, k_3 and k_4 for each of the four equations, i.e. $\mathbf{k}_1 = (k_{11}, k_{12}, \ldots, k_{14})$, and similarly for $\mathbf{k}_2$, $\mathbf{k}_3$ and $\mathbf{k}_4$. To implement this in a computer program, at each step you must first compute all of the components of the vector $\mathbf{k}_1$, then use this to get the components of $\mathbf{k}_2$, all of which are needed to get $\mathbf{k}_3$, and so forth.

4 Teamwork

You will need to divide up the work so that each member has specific tasks to accomplish and the team achieves its goals to the fullest extent possible. Below are some suggestions on possible ways of separating the project into individual tasks.

In Java you will probably define a class called, say, `FourVector`, that represents the solution vector $\mathbf{u} = (x, y, v_x, v_y)$. Your program should create an object of type `FourVector` and initialise it with the starting conditions $\mathbf{u}(t_0) = \mathbf{u}_0$. You should also define a class which has methods to evaluate the derivatives of $\mathbf{u}$. In addition you should:

- Implement an `Euler` class with a method that carries out the updating rule described by equations (10)–(13).
- Implement an `RK` class with a method that carries out the updating rule described by equations (27). This method should have the same name and return types as the corresponding method in the `Euler` class so that they can be used interchangeably.
- Store the data values produced by successive updates so you can pass them to a plotting routine.

You should write a driver program with the `main` method that creates objects using the classes above and executes the various methods in the desired way.

5 References

The Runge–Kutta method is discussed in most books on differential equations. A complete description can be found in

W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes*, 2nd edition, Cambridge University Press, Cambridge (1992).

Numerical solution of the Kepler problem is discussed in

N.J. Giordano, *Computational Physics*, Prentice Hall (1997).

G. Cowan, 19 November 2001