Numerical Methods in Physics Week 1

Nikolai Plambech Nielsen

1 Nuclear Decay

Suppose two radioactive isotopes, A and B, are present with populations $N_A(t)$ and $N_B(t)$, and decay times τ_A and τ_B. Type A decays into type B, while B decays into something not tracked. The relevant differential equations are

$$\frac{\mathrm{d}N_A}{\mathrm{d}t} = -\frac{N_A}{\tau_A}, \quad \frac{\mathrm{d}N_B}{\mathrm{d}t} = \frac{N_A}{\tau_A} - \frac{N_B}{\tau_B}. \tag{1.1}$$

The purpose of this assignment is to solve this problem numerically for a number of different conditions, using Euler integration. This method is used for its simplicity and ease of implementation. For all of the following simulations and plots, the value $\Delta t = 0.1$ is used.

1.1 Compare the numerical and analytical solutions

The analytical solutions are given in the assignment, and are as follows:

$$N_A(t) = N_A(0) \exp\left(-\frac{t}{\tau_A}\right), \tag{1.2}$$

$$N_B(t) = \begin{cases} N_B(0)\exp\left(-\frac{t}{\tau_A}\right) + t\frac{N_A(0)}{\tau_A}\exp\left(-\frac{t}{\tau_A}\right), & \tau_A = \tau_B, \\ N_B(0)\exp\left(-\frac{t}{\tau_B}\right) + \frac{N_A(0)}{\frac{\tau_A}{\tau_B}-1}\left[\exp\left(-\frac{t}{\tau_A}\right) - \exp\left(-\frac{t}{\tau_B}\right)\right], & \tau_A \neq \tau_B. \end{cases} \tag{1.3}$$

The results, along with residual plots, for $N_A(0) = N_B(0) = 1000$ and $\tau_A = 5, \tau_B = 10$ are shown below, with the analytical solution for $\tau_A \neq \tau_B$:
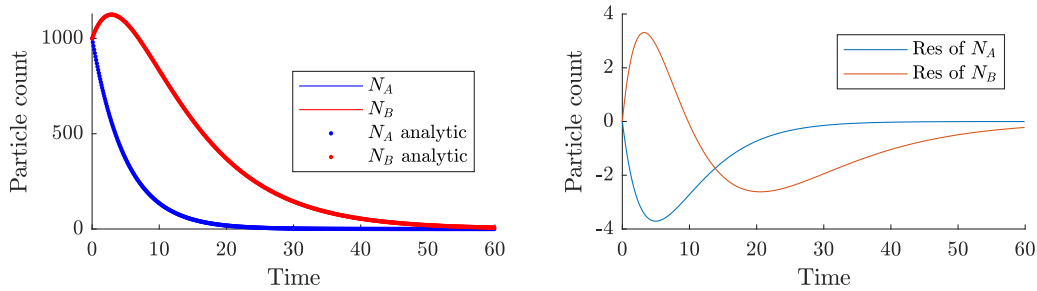

Figure 1: Two plots of nuclear decay as a function of time. In the left plot, the numerical and analytical solutions are shown on the same plot, while in the right plot, the residuals of N_A and N_B are shown to highlight the discrepancy between the analytical and numerical solutions to the problem.

To demonstrate the solution for $\tau_A = \tau_B$, the initial conditions of $N_A = N_B = 1000$ and $\tau_A = \tau_B = 10$ are shown below. Again with accompanying residual plots:
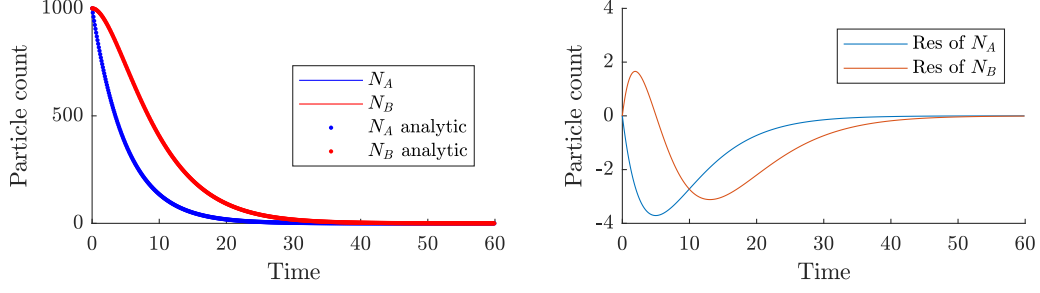
Figure 2: Same plot setup as in figure 1, but with equal values for τ_A and τ_B.

As seen in both figures 1 and 2, the numerical is close to the analytical solution, differing by less than 4 at all times, for each of the cases. An increase in N_B is seen in figure 1 due to the faster decay of type A than that of type B.

1.2 Explain the limit of $\tau_A/\tau_B \gg 1$

In this limit, the decay of type B is much faster than that of type A. This means, that on any appreciable time scale of change for N_A, the population of B reaches a steady-state solution, where $\mathrm{d}N_B/\mathrm{d}t = 0$. As such the differential equation for N_B becomes

$$\frac{\mathrm{d}N_B}{\mathrm{d}t} = \frac{N_A}{\tau_A} - \frac{N_B}{\tau_B} = 0, \quad \Rightarrow \quad N_B(t) = N_A\frac{\tau_B}{\tau_A}. \tag{1.4}$$

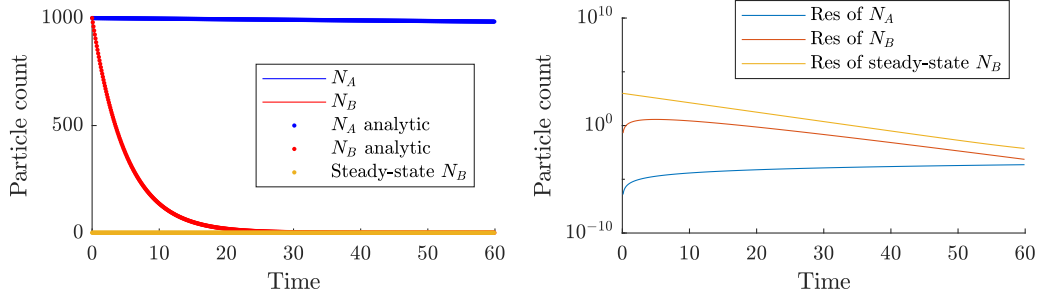This is also seen in a simulation, where $\tau_A = 3600, \tau_B = 5$:

Figure 3: Plot of the numerical and analytical solutions to the decay problem (left) and associated residuals (right). In both cases, the steady-state solution for N_B is also plotted.

as seen, the steady-state solution for N_B ((1.4)) does correctly predict the behaviour for N_B, when this is approximately 0. This makes sense; the change in N_A is almost constant, corresponding to when N_B is approximately 0, which is the steady state for this particular problem.

2 Projectile motion

For this assignment, projectile motion is to be simulated, again using the Euler-method. This gives another example of how to use this simple method of simulation, for a problem which (without wind resistance) has an analytical solution. A drag force is introduced after this, to give an example of a numerical problem with no analytical solution.

Furthermore, for this problem, a second derivative is used, whereby a second Euler integration is needed, to complete the time step. The relevant differential equations are

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \mathbf{v}, \quad \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -g\hat{\mathbf{y}}. \tag{2.1}$$

where the analytical solution is computed by integrating the second equation twice, and using the relevant initial conditions:

$$\mathbf{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_0 + v_{x,0}t \\ y_0 + v_{y,0}t - gt^2/2 \end{pmatrix}. \tag{2.2}$$

2.1 Without wind resistance

The numerical and analytical solutions to projectile motion, given the initial conditions of $x_0 = 0, y_0 = 2$ m, $v_0 = 4$ m/s and $\theta = 70°$, where $v_{x,0} = v_0 \cos\theta$ and $v_{y,0} = v_0 \sin\theta$. For the numerical solution a value of $\Delta t = 0.01$ s is used. The results are shown below, along with the absolute residual, as a function of time:
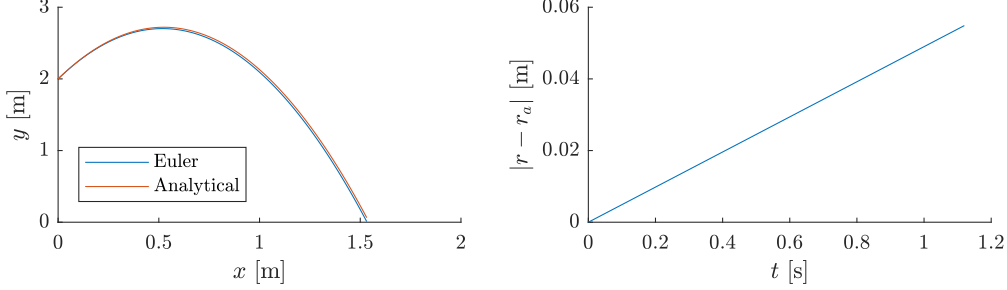

Figure 4: Plots of the numerical and analytical solutions to projectile motion (left) and the absolute residual (right).

The absolute residual is also the global truncation error, as defined in the slides for the week 1 lectures. As is expected, the error increases linearly in time, given the linear increase in velocity. This means that the Euler integration underestimates the velocity by a constant amount per time step, leading to a linear increase in global truncation error.

Next the effect of the size of the time step Δt is analysed. This is done by creating a vector of logarithmically spaced values for Δt, between 10^{-1} s and 10^{-8} s, running the simulation, and recording the global truncation error:
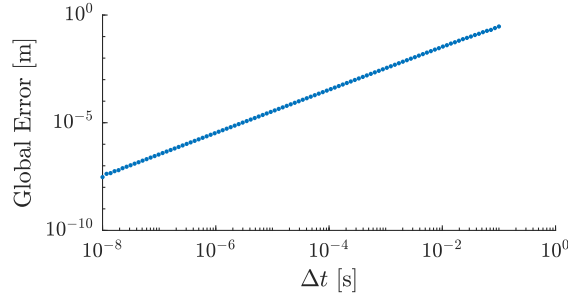

Figure 5: A plot of the global error of the simulation, as a function of the time step size Δt.

On the double-log plot, the relation between global truncation error and time step size is definitely polynomial, as the graph is that of a straight line. The slope then corresponds to the polynomial degree, and as this is approximately 1, the relation is taken as being linear.

2.2 With wind resistance

Next a drag force, quadratic in velocity, is introduced to the problem. The formula is

$$\mathbf{F}_d = -\frac{1}{2}C_D \rho A v \mathbf{v} \tag{2.3}$$

where C_D is the coefficient of drag for the projectile, ρ is the density of the surrounding media, A is the cross sectional area of the projectile, and v is the magnitude of the velocity vector $\mathbf{v}$. As seen, this is directed opposite to the velocity. The differential equations become:

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}, \quad \frac{d\mathbf{v}}{dt} = -\frac{1}{2m}C_D \rho A v \mathbf{v} - g\hat{\mathbf{y}}. \tag{2.4}$$

where now the second differential equation is no longer linear, owing to the quadratic term in velocity.

With no analytical solution given, other means for estimating the error has to be utilized. As confirmed in the example with no wind resistance, the global truncation error is linearly related to the size of the time step Δt. As such, the global truncation error on the simulation with wind resistance will also approach 0 as the size of Δt is decreased. Given this, a method of attaining a desired level of precision is needed.

For the problem at hand, the simulation is run with 100 logarithmically spaced values between 10^{-1} s and 10^{-8} s. The final point of impact is logged, along with the time taken to run the simulation. The results are shown below on semilog and double-log plots respectively:
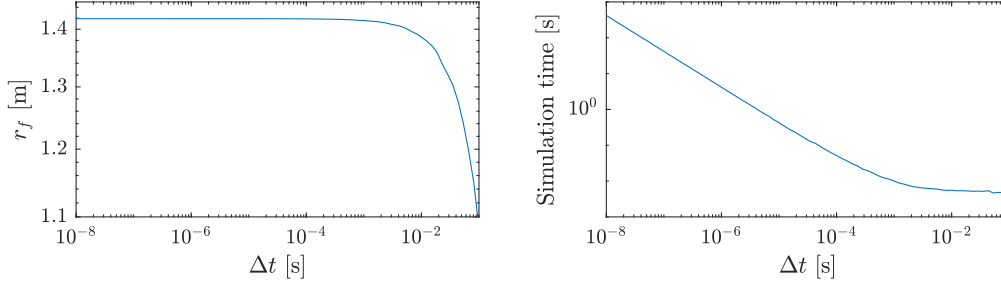

Figure 6: Semilog plot of the final distance (left) and double-log plot of the simulation time (right). Both as a function of Δt.

The reason for also recording the simulation time is as follows: The final simulation, with $\Delta t = 10^{-8}$ s, took just under 7 minutes, but running a simulation, where the difference between the final results of this and the final simulation is 1% took only 0.0061 seconds. This simulation has a step size of $\Delta t = 0.0043s$. So while the precision is within 1 % of the final simulation, the run time is almost 5 orders of magnitude smaller. The desired precision can of course be chosen as needed, achieving a precise enough results whilst keeping run time small enough to be practical.

3 Orbits of comets

For this assignment, the trajectory of particles in a gravitational field is calculated both using Euler integration and 4th order Runge-Kutta integration (RK4), which uses a weighted average of several derivatives with smaller time steps, to calculate the final value of the next time step. The differential equations for this problem is

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}, \quad \frac{d\mathbf{v}}{dt} = \frac{GM}{r^2}\hat{\mathbf{r}} = \frac{GM}{r^3}\mathbf{r}, \tag{3.1}$$

where, for the purpose of simplicity, GM is set to $4\pi^2$ $(\text{AU})^3/(\text{yr})^3$. This gives a circular orbit for an initial radius of 1 AU and initial tangential velocity of 2π AU/ yr. Furthermore, the mass of the comet in orbit, is set to unity.

3.1 Euler and RK4

First, the two methods are compared for a circular orbit with a step size of $\Delta t = 0.02$ yr and a total simulation time of 1 yr. The trajectories are shown in the figure below:
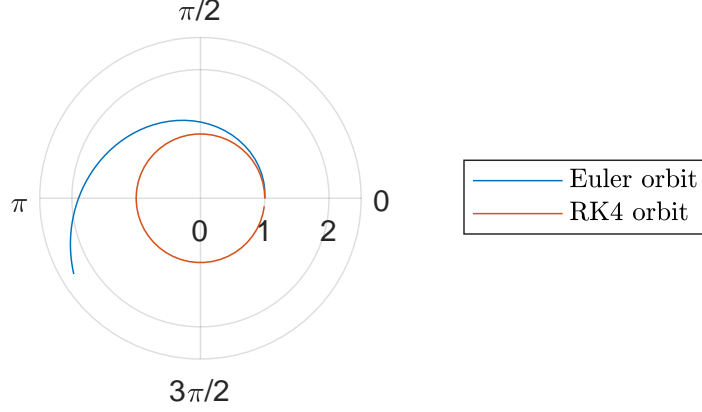
Figure 7: Simulation of a circular orbit using both the Euler method and the 4th order Runge-Kutta method.

As seen, neither of the simulations actually complete an orbit, however the Runge-Kutta method comes significantly closer. To quantify this, the following fractional error is calculated:

$$\text{err} = \frac{|\mathbf{r}_f - \mathbf{r}_0|}{|\mathbf{r}_0|} \tag{3.2}$$

where $\mathbf{r}_0$ is the initial and $\mathbf{r}_f$ is the final position of the comet. Next, the simulation is run for 60 different values of Δt, logarithmically spaced between 10^{-1} yr and 10^{-6} yr:
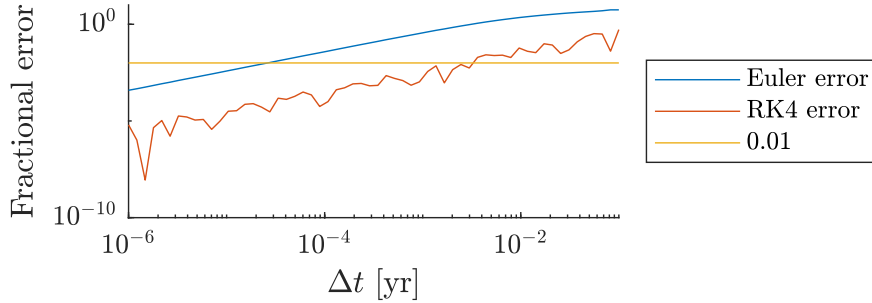

Figure 8: A plot of the global fractional error as a function of the time step size Δt, for both the Euler and Runge-Kutta methods. A horizontal line for 1% is included for clarity.

As seen from the figure, the Runge-Kutta method consistently outperforms the Euler method in fractional error, though the error in the Runge-Kutta method fluctuates, sometimes over an order of magnitude. Rounded up, the Runge-Kutta needs a Δt of the order 10^{-3} yr to have a fractional error lower than 1%, whilst the Euler method needs one in the order of 10^{-5} yr, 2 orders of magnitude lower.

3.2   RK4 and the analytical solution

For this problem, the analytical solution is also known, and the radius, as a function of the angle θ, is given by

$$r(\theta) = \frac{a(1 - \epsilon^2)}{1 - \epsilon \cos \theta} \tag{3.3}$$

where $\epsilon = \sqrt{1 - b^2/a^2}$ is the eccentricity of the orbit, $a = (r_{\max} + r_{\min})/2$ is the semimajor axis and $b = \sqrt{r_{\max} r_{\min}}$ is the semiminor axis of the orbit.

Next the Runge-Kutta method is compared to the analytical solution. The smallest and largest radii of the orbit are calculated after the full simulation has run, and the magnitude of the radius is calculated at each point along the orbit. To compare the two solutions, two initial conditions are chosen: both starting at a radial distance of 1 AU, with initial tangential velocity of π AU/yr and 2π AU/yr. Both simulations are run with a step size of $\Delta t = 0.005$ yr.

With the angle, semimajor and semiminor axes calculated, the analytical solution can be computed. The numerical solution is then compared by computing the absolute fractional error across the entire period of an orbit:

$$\text{err} = \left| \frac{r_{\text{numerical}} - r_{\text{analytical}}}{r_{\text{analytical}}} \right| \tag{3.4}$$

The results, for the two initial conditions are shown in the figures below:
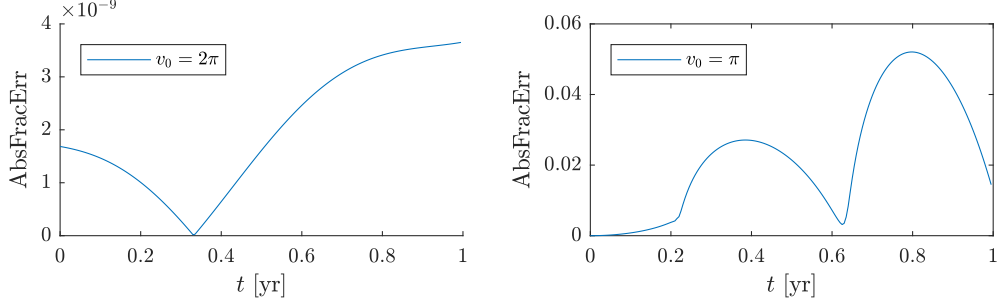

Figure 9: Plots of the absolute fractional error between the analytical and numerical solution to the Kepler problem. On the left for a circular orbit and on the right for an elliptical orbit.

For the circular orbit the absolute fractional error is exceedingly small, 9 orders of magnitude below unity, whilst for the elliptical orbit it is in the small percentages, between 0 and 6 percent. The reason for the dips in fractional error is seen in figure 11, as the Runge-Kutta method slowly circles inwards but at the smallest radius, the orbits align again.

3.3 Adaptive 4th order Runge-Kutta

For this last exercise, an adaptive version of the 4th order Runge-Kutta method is introduced. This performs the integration three times. Once normally, and then twice in a row, with half the time step size, to get one full integration with the chosen step size. The big step is called $x_b(t + \Delta t)$ and the result of the small step(s) is called $x_s(t_+\Delta t)$. If the error, defined as err $= |x_b(t + \Delta t) - x_s(t + \Delta t)|$, is smaller than some specified upper bound, the time step size Δt is accepted and increased for the next time step, but if the error is *larger* than the upper bound, the step size is rejected, decreased, and the error is calculated again with the smaller step size.

This allows the simulation to have a starting size for Δt, but to increase it when the function is only changing slowly, and decreasing it when the function is changing rapidly.

For the same initial conditions as before, when comparing the analytical and regular Runge-Kutta method, the adaptive and non-adaptive Runge-Kutta methods are compared, by calculating the trajectories and evolution of energy, as a function of time.
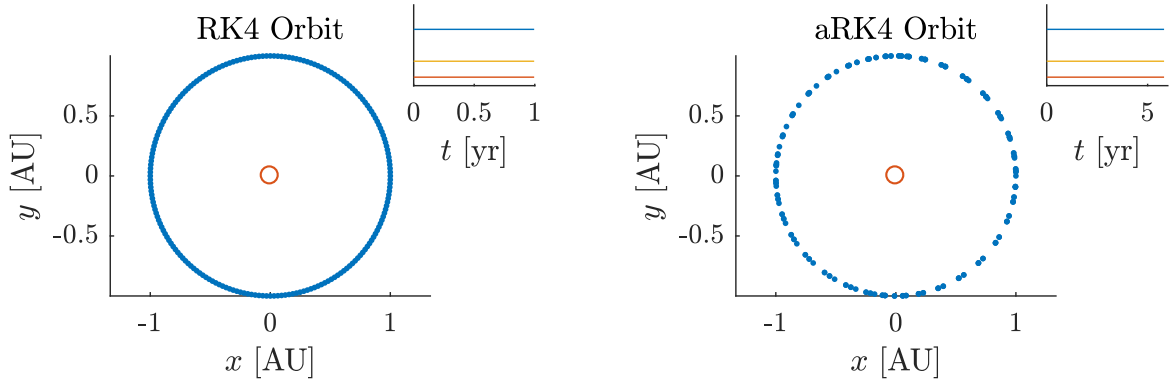

Figure 10: Plots of the trajectory of a circular orbit, along with inset plots of the kinetic (blue), potential (red), and total energy (orange), as a function of time, for the Runge-Kutta (left) and adaptive Runge-Kutta (right) methods.
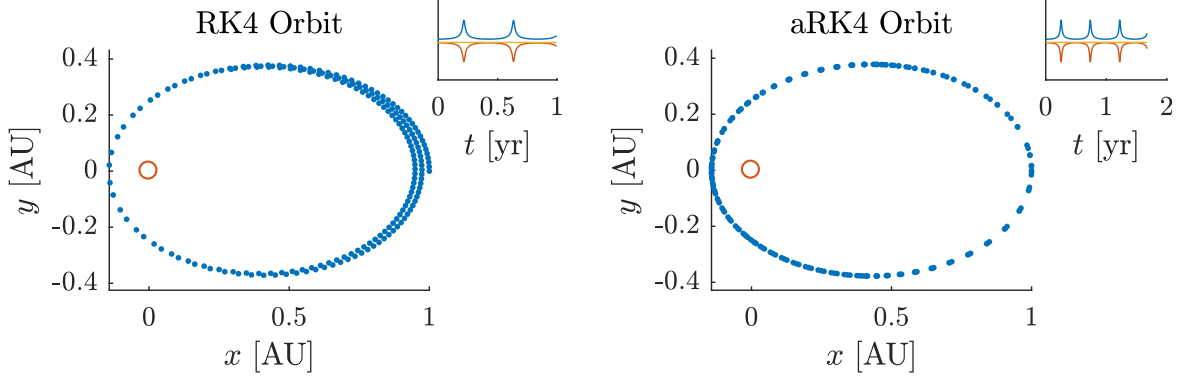
6

Figure 11: Same plot setup as before, but for an elliptical orbit

Furthermore to see the adaptiveness in action, the step size Δt is plotted as a function of the radial distance:
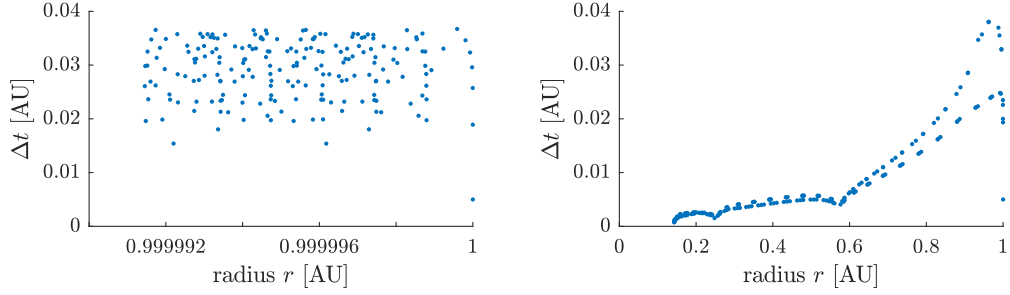

Figure 12: Plots of the time step size Δt, as a function of radial distance, for a circular (left) and elliptical (right) orbit.

As seen in figure 10, both methods accurately simulate the circular orbit. However, with the same number of iterations, the adaptive method completes almost 6 full orbits, whilst the non-adaptive completes only one. This shows that the initial step size was small enough, that an increase in its size did not result in any significant errors. For the elliptical orbit, however, the non-adaptive method performs significantly worse than the adaptive one. The reason for this is seen in figure 12, where it is seen that the step size around the smallest radius is smaller than the initial value of Δt. This suggests that for the non-adaptive method to produce accurate results, the step size should be reduced.

The step sizes shown in figure 12 shows the adaptiveness at work. In the circular orbit the size of the time step oscillates as the error oscillates around the upper bound set. For the elliptical orbit, the step size decreases as the radius decreases. This makes sense, as the velocity increases when the radius decreases, which necessitates a smaller time step, if the error bound is to be satisfied. In both cases the initial step size is well within the upper bound, and the size is quickly increased.