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 values $\Delta t = 0.1$ 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 $\tau_A \neq \tau_B$:

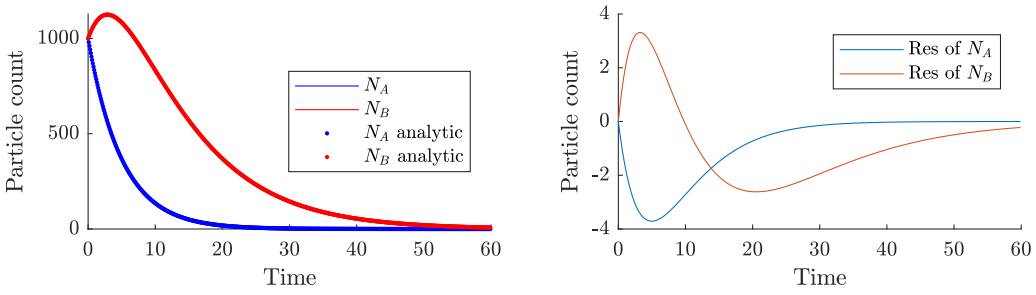

Figure 1: **INSERT PURDY CAPTION PL0X**

COMMENT ON RESULT, YAS, V V V GUT 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:

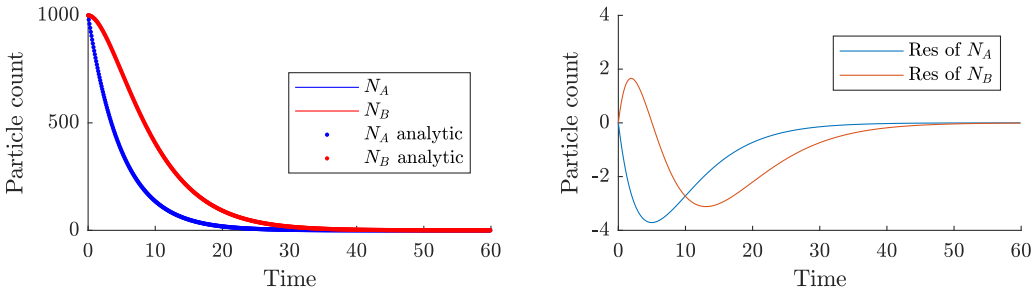

Figure 2: **INSERT PURDY CAPTION PL0X**

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 $\tau_A = 3600, \tau_B = 5$:

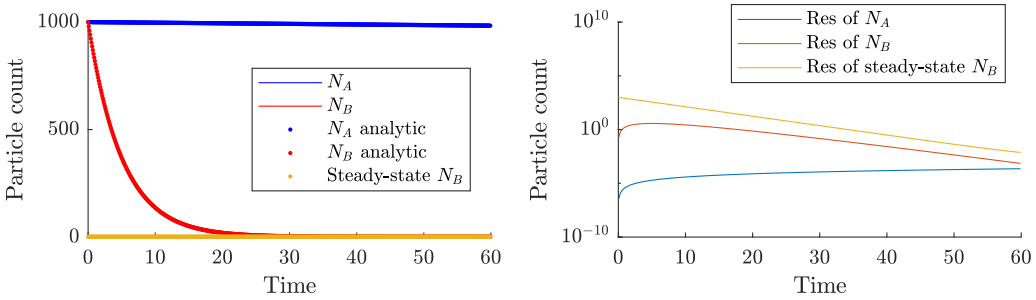

Figure 3: **INSERT PURDY CAPTION PL0X**

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. After this, uncharted waters are encountered, when wind resistance is included. Here no analytical solution is given, and all reliance upon the previous methods, like residuals between the numerical and analytical solutions, are not applicable.

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:

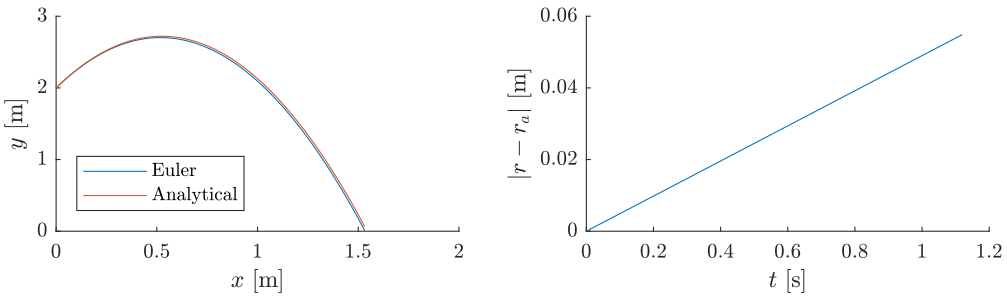

Figure 4: **INSERT PURDY CAPTION PL0X**

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. If any smaller step size is to be investigated, the simulation as to be broken into pieces, as the size of the arrays exceed the amount of memory available on the desktop workstation used.

This could be done by recording the first, say, 10^6 steps, storing the final results, clearing the arrays from memory and then running the simulation again, with the final results as the new initial conditions, until the full simulation has been completed.

However, with the step sizes as mentioned, the final global truncation error becomes

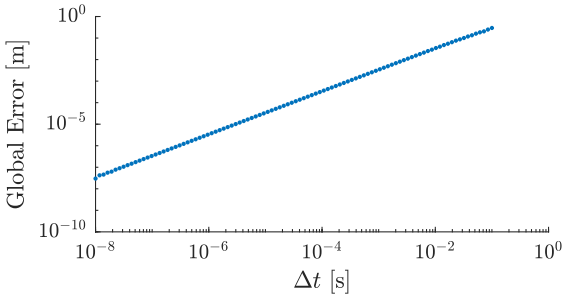

Figure 5: **INSERT PURDY CAPTION PL0X**

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 **REMEMBER CHECKUP**, 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:

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

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:

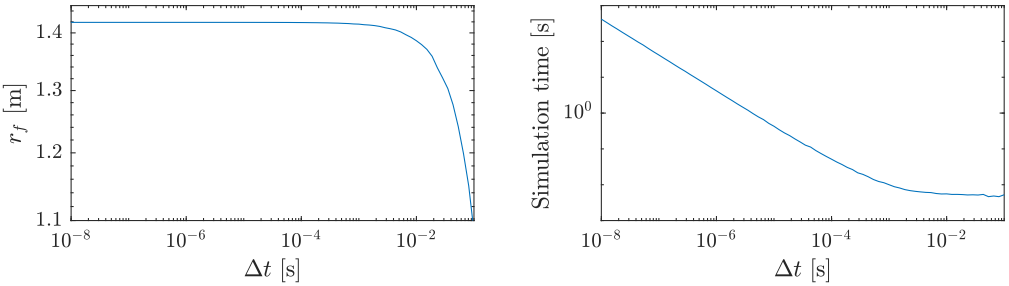

Figure 6: **INSERT PURDY CAPTION PL0X**

the reason for also recording the simulation time is that while a simulation taking a week to run might be the most precise result achievable, it is not very practical, and might not be that much more precise than that of a simulation that only takes a minute or two. For example, the final simulation, with $\Delta t = 10^{-8}$ s, took just under 7 minutes. The simulation with the smallest Δt, coming within 1% of the result of the final simulation took just 0.0061 seconds, with a time 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, 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

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

where, for the purpose of simplicity, GM is set to $4\pi^2$ $(\mathrm{AU})^3/\ (\mathrm{yr})^3$. 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:

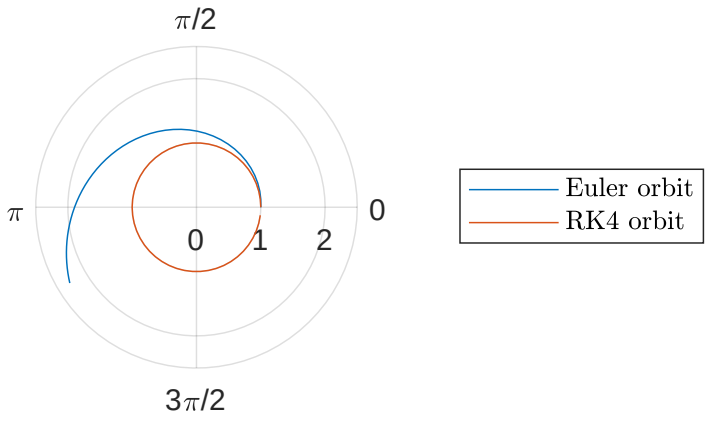

Figure 7: **INSERT PURDY TEXT**

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:

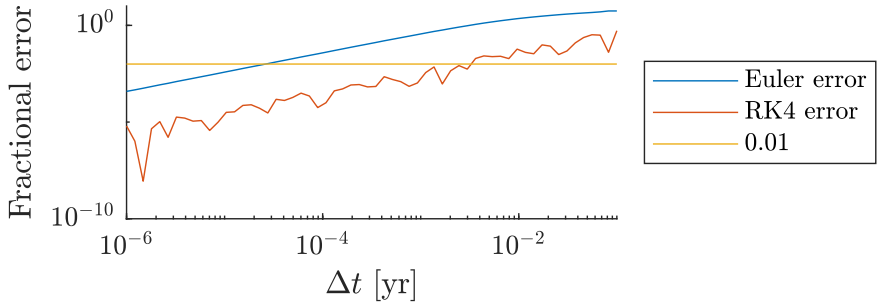

Figure 8: **INSERT PURDY TEXT**

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 is the semimajor axis and b is the semiminor axis of the orbit. The semimajor and semiminor axes are given respectivle by the arithmetic and geometric mean of the largest and smallest radius:

$$a = \frac{r_{\max} + r_{\min}}{2}, \quad b = \sqrt{r_{\max} r_{\min}}. \tag{3.4}$$

Next the RK4 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.

The simulation uses cartesian coordinates, so to retrieve the angle used in the analytical solution the function `atan2` is used. This is different to the normal arctangent function in that `atan2` takes two inputs, y and x, whereas the normal arctangent only takes one $v = y/x$. The normal arctangent function is only defined for angles between $-\pi/2$ and $\pi/2$ radians, so in the right half-plane. But the `atan2` function works for all angles between 0 and 2π radians, and as such is the one needed for this problem.

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:

$$\text{err} = \left| \frac{r_{\text{numerical}} - r_{\text{analytical}}}{r_{\text{analytical}}} \right| \tag{3.5}$$

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

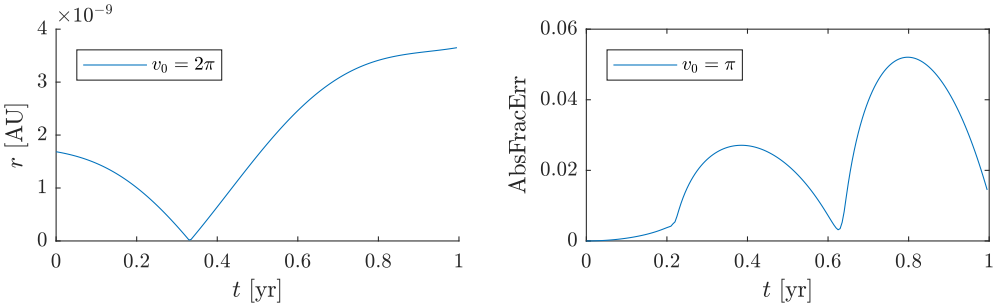

Figure 9: **INSERT PURDY TEXT**

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 $\text{err} = |x_b(t+\Delta t) - x_s(t+\Delta t)|$, 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.

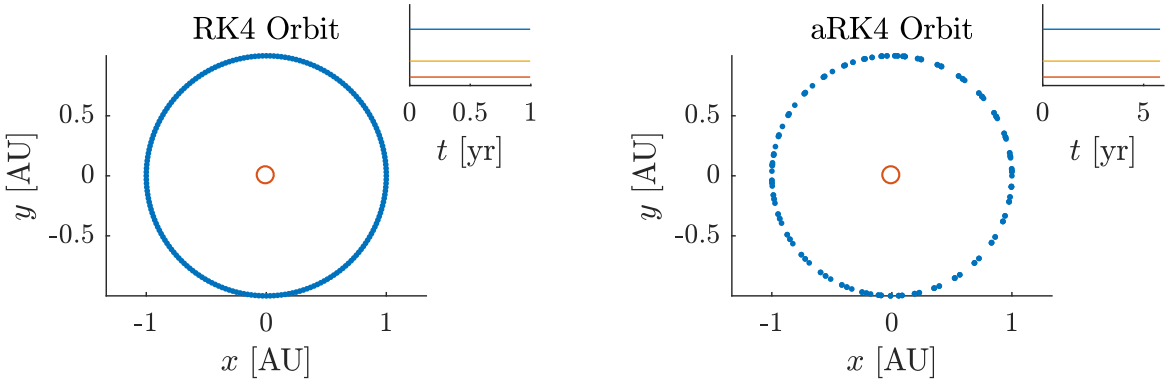

Figure 10: **INSERT PURDY TEXT**

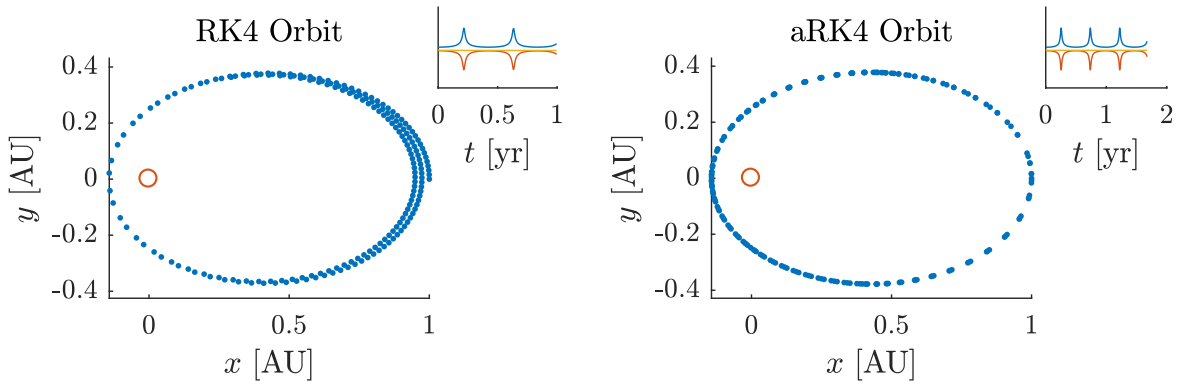

Figure 11: **INSERT PURDY TEXT**

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

Figure 12: **INSERT PURDY TEXT**