4\textsuperscript{th} order Runge-Kutta method with fixed step size

The most widely used fixed step-size Runge Kutta method is of 4\textsuperscript{th} order

Let
\begin{align*}
k_1 &= \Delta t \ f(t(i), y(t(i))) \\
k_2 &= \Delta t \ f(t(i) + \frac{1}{2} \Delta t, y(t(i)) + \frac{1}{2} k_1) \\
k_3 &= \Delta t \ f(t(i) + \frac{1}{2} \Delta t, y(t(i)) + \frac{1}{2} k_2) \\
k_4 &= \Delta t \ f(t(i) + \Delta t, y(t(i)) + k_3)
\end{align*}

2nd - order Runge-Kutta method
\[ y(i + 1) = y(i) + k_2 + O(\Delta t^3) \]

4th - order Runge-Kutta method
\[ y(i + 1) = y(i) + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) + O(\Delta t^5) \]

Note: when you use a fixed step-size method, always make sure in the end that your results do not change significantly with step size.

Note: going to higher order and smaller step-sizes to improve accuracy can be very time consuming and may not be the best approach → investigate the source of the problem
Adaptive step-size methods

Consider a function $y(t)$ like this:

It is desirable to have: (why?)

- small steps in the region of large variation
- large steps in the region of small variation

$\Rightarrow$ adapt the step size of a Runge-Kutta method as you are calculating the solution

Idea of a 4th/5th order method:

For a given step size $\Delta t$
- do a single step in 5th order approximation $\Rightarrow y(i+1)$
- do the same step in 4th order approximation $\Rightarrow y^*(i+1)$

Then $\Delta y = |y(i+1) - y^*(i+1)|$ is a measure for the error, which is $O(\Delta t)^5$

Hence, we can determine a new step size $\Delta t$ that matches our tolerance for $\Delta y$
- if $\Delta y$ is too large $\Rightarrow$ reduce the step size $\Delta t$ and try again
- if $\Delta y$ is too small $\Rightarrow$ increase the step size $\Delta t$ and try again
There are several ways to specify the tolerance:

1. by relative error
   When would this be a problem? \[ \left| \frac{\Delta y}{y(i)} \right| < \epsilon_{\text{relative}} = \text{RelTol} \text{ in Matlab} \]

2. by absolute error
   What has to be considered here? \[ |\Delta y| < \epsilon_{\text{absolute}} = \text{AbsTol} \text{ in Matlab} \]

3. by cumulative error
   - if conserved quantities are known (for example energy or momentum) one can adjust the step size to keep the conserved quantities within tolerance
   - this requires special programming for each particular problem, but is very powerful when available

Matlab provides two ODE solvers based on adaptive step-size Runge-Kutta methods ode23 (2\(^{\text{nd}}/3\(^{\text{rd}}\) order method) and ode45 (4\(^{\text{th}}/5\(^{\text{th}}\) order method)

% calculate a solution to the nuclear decay problem from the built-in ode45 solver
options=odeset('RelTol',1.e-6,'AbsTol',1.e-6);
[tt,yode45]=ode45('f1nuc',t,y0,options);
solution rhs of times initial
vector y(t) \( dy/dt \) vector condition