Out: February 24, 2020
Due: March 8, 2020

NTNU – Trondheim
Norwegian University of
Science and Technology

TTK4130 Modeling and Simulation
Assignment 7

Introduction

The objective of this assignment are:

- To understand how explicit Runge-Kutta (ERK) methods work, and to be able to implement them.
- To understand important concepts of ERK methods, such as accuracy and stability, and the related concepts of local and global error, as well as stability functions.
- To be able to explain how an adaptive integrator works.

Moreover, this is the first assignment where we are going to implement our own differential equation solver. The work done here will be the cornerstone for the solvers developed in assignment 8 and 9. In order to use and implement these solvers, it will be very advantageous to understand how one can convert symbolic expression into function handles or files with `matlabFunction`, and how one can create new function handles using the `@` sign.

Problem 1 (Explicit Runge-Kutta 2 methods)

For a dynamic model:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, t\right), \tag{1}$$

where for the sake of simplicity we assume the input $\boldsymbol{u} = \boldsymbol{u}_k$ to be constant on the time interval $[t_k, t_{k+1}]$, ERK2 methods are based on 2 evaluations of the function $\boldsymbol{f}$, the first of which occurs at $[\boldsymbol{x}(t_k), \boldsymbol{u}_k]$. They can generally be written as

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), t_k\right) \tag{2a}$$
$$\boldsymbol{k}_2 = \boldsymbol{f}\left(\boldsymbol{x}(t_k) + a \cdot \Delta t \cdot \boldsymbol{k}_1, \boldsymbol{u}(t_k + c\Delta t), t_k + c\Delta t\right) \tag{2b}$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}\left(t_k\right) + \Delta t \sum_{i=1}^{2} b_i \cdot \boldsymbol{k}_i. \tag{2c}$$

and have the Butcher tableau

$$\begin{array}{c|cc} 0 & & \\ \mathrm{c} & a & \\ \hline & b_1 & b_2 \end{array}$$

(Zeros in the tableau are traditionally omitted)

The ERK2 method will have a numerical local error $e_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}\left(t_{k+1}|k\right)$, where $\boldsymbol{x}(t|k)$ is the actual trajectory of the ODE with the initial condition $\boldsymbol{x}(t_k|k) = \boldsymbol{x}_k$.

(a) For the sake of simplicity, consider the case where $\boldsymbol{f}$ is time-independent (t does not enter as an argument) and $\boldsymbol{u}(t) = \boldsymbol{u}_k$ is constant over the time interval $[t_k, t_{k+1}]$.

Provide conditions on $a, b_{1,2}$ and c such that the error e_k of the method is of order 3.

Hint: Observe that

$$\boldsymbol{x}\left(t_{k+1}\right) = \boldsymbol{x}\left(t_k\right) + \Delta t \cdot \boldsymbol{f}\left(\boldsymbol{x}(t_k), \boldsymbol{u}_k\right) + \frac{\Delta t^2}{2} \cdot \dot{\boldsymbol{f}}\left(\boldsymbol{x}(t_k), \boldsymbol{u}_k\right) + \mathcal{O}\left(\Delta t^3\right). \tag{3}$$

Solution: First, note that the definition of an ERK method implies that $a = c$.
We observe that

$$\dot{\boldsymbol{f}}\left(\boldsymbol{x}(t_k), \boldsymbol{u}\left(t_k\right)\right) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{f}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)} + \frac{\partial \boldsymbol{f}}{\partial t}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)}$$

and

$$\begin{aligned}
\boldsymbol{k}_2 &= \boldsymbol{k}_1 + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)} a \cdot \Delta t \cdot \boldsymbol{k}_1 + \frac{\partial \boldsymbol{f}}{\partial t}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)} c \cdot \Delta t + \mathcal{O}\left(\Delta t^2\right) \\
&= \boldsymbol{k}_1 + a \cdot \Delta t \cdot \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{f}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)} + \frac{\partial \boldsymbol{f}}{\partial t}\bigg|_{\boldsymbol{x}(t_k),\boldsymbol{u}(t_k)} \right) + \mathcal{O}\left(\Delta t^2\right) \\
&= \boldsymbol{k}_1 + a \cdot \Delta t \cdot \dot{\boldsymbol{f}}\left(\boldsymbol{x}(t_k), \boldsymbol{u}\left(t_k\right)\right) + \mathcal{O}\left(\Delta t^2\right),
\end{aligned}$$

Hence, by using (2c) we have that

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}\left(t_k\right) + \Delta t b_1 \boldsymbol{f}\left(\boldsymbol{x}(t_k), \boldsymbol{u}_k\right) + \Delta t b_2 \boldsymbol{f}\left(\boldsymbol{x}(t_k), \boldsymbol{u}_k\right) + a \cdot b_2 \cdot \Delta t^2 \cdot \dot{\boldsymbol{f}}\left(\boldsymbol{x}(t_k), \boldsymbol{u}\left(t_k\right)\right) + \mathcal{O}\left(\Delta t^3\right).$$

The identification of this equation and (3) then yields

$$b_1 + b_2 = 1, \qquad a \cdot b_2 = \frac{1}{2}.$$

(b) ERK2 methods are at best of order 2. How does that relate to $e_k = \mathcal{O}\left(\Delta t^3\right)$?

Solution: The order of a RK method relates to the global error, while e_k is the local error.
The one-step error e_k is at best of order 3. In order to simulate the dynamics (1) over a time interval $[0, t_\mathrm{f}]$, we need $N = t_\mathrm{f}/\Delta t$ steps. Each of these steps yields an error of order 3, which potentially accumulates over the integration. It follows then that the global integration error is

$$e_{\text{Global}} = \mathcal{O}\left(N\Delta t^3\right) = \mathcal{O}\left(t_\mathrm{f}\Delta t^2\right).$$

Therefore, the ERK2 methods are at best of order 2.

(c) **(Optional)** Consider a system having a trajectory given by a polynomial of order n, i.e.:

$$\boldsymbol{x}\left(t\right) = \boldsymbol{x}(t_k) + \sum_{i=1}^{n} \frac{\alpha_i}{i!}\left(t - t_k\right)^i, \quad t \in [t_k,\, t_{k+1}]. \tag{4}$$

For what values of n, $b_{1,2}$, and c is the ERK2 method exact on (4)?
What do you observe from your computations?

Solution: We observe that

$$\boldsymbol{x}\left(t_{k+1}\right) = \boldsymbol{x}_k + \sum_{i=1}^{n} \frac{\alpha_i}{i!}\Delta t^i$$

and

$$\boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) = \sum_{i=1}^{n} \frac{\alpha_i}{(i-1)!}\left(t - t_k\right)^{i-1}, \quad t \in [t_k,\, t_{k+1}].$$

Moreover,

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}\left(\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), t_k\right) = \alpha_1 \\
\boldsymbol{k}_2 &= \boldsymbol{f}\left(\boldsymbol{x}(t_k) + a \cdot \Delta t \cdot \boldsymbol{k}_1, \boldsymbol{u}(t_k + c \cdot \Delta t), t_k + c \cdot \Delta t\right) = \sum_{i=1}^{n} \frac{\alpha_i}{(i-1)!} c^{i-1} \Delta t^{i-1} \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + b_1 \alpha_1 \Delta t + b_2 \sum_{i=1}^{n} \frac{\alpha_i}{(i-1)!} c^{i-1} \Delta t^{i}.
\end{aligned}$$

We then identify the terms for Δt^i in the expressions for $\boldsymbol{x}\left(t_{k+1}\right)$ and $\boldsymbol{x}_{k+1}$. This gives:

$$\begin{aligned}
i = 1 \quad \text{yields:} \quad & b_1 \alpha_1 + b_2 \alpha_1 = \alpha_1 \qquad \Rightarrow \qquad b_1 + b_2 = 1\,, \\
i > 1 \quad \text{yields:} \quad & \frac{\alpha_i}{i!} = b_2 \frac{\alpha_i}{(i-1)!} c^{i-1} \qquad \Rightarrow \qquad \frac{1}{i} = b_2 c^{i-1}.
\end{aligned}$$

The second condition is satisfied for $i = 2$ when

$$c \cdot b_2 = \frac{1}{2}.$$

For $i > 2$, the equality cannot match anymore, such that $n = 2$ is the maximum degree for which the numerical integration of the polynomials can be exact.
We observe that

- The conditions on $b_{1,2}$ and c are identical to the conditions found in part a. These conditions make the ERK2 method of order 2.
- The largest polynomial order that can be integrated exactly by a ERK2 is $n = 2$, i.e. it matches the order of the method. This observation is actually general for ERK methods.

Problem 2 (Accuracy and stability)

We will start by investigating the accuracy and computational cost of some explicit integration schemes. We will consider the explicit Euler scheme and other 2 explicit Runge-Kutta scheme of order 2 and 4, respectively. The Butcher tables of these three schemes are:

Table 1: Explicit Euler (RK1)

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

Table 2: Runge-Kutta 2 (RK2)

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

Table 3: Runge-Kutta 4 (RK4)

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & & \frac{1}{2} & & \\ 1 & & & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

We will use these methods to simulate the classic test system:

$$\dot{x} = \lambda x\,, \tag{7}$$

where $\lambda < 0$.

(a) Implement the 3 ERK schemes for a generic vector field $\boldsymbol{f}(t, \boldsymbol{x})$ (i.e. $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x})$).
Test the implemented codes for $\lambda = -2$ in the time interval $[0, 2]$. Choose a time-step $\Delta t = 0.4$ and the initial condition $x(0) = 1$. Comment on the results.

Hint: It is recommended to code a Matlab function that implements an arbitrary ERK method based on its Butcher table. The file `ERKTemplate.m` *provides a template for such a function. In addition, you can use the routine* `TestERK.m` *to test your implementation. This routine should simulate the classical mass-damper-spring system.*

Solution: An example of Matlab function that implements an arbitrary ERK method is

```
function x = ERK(ButcherArray, f, T, x0)
    Nt = length(T);
    Nx = length(x0);
    dT = diff(T);
    x = zeros(Nx,Nt);
    x(:,1) = x0;
    A = ButcherArray.A;
    b = ButcherArray.b(:);
    c = ButcherArray.c(:);
    Nstage = size(A,1);
    K = zeros(Nx,Nstage);
    xt = x0;
    for nt=2:Nt
        t = T(nt-1);
        dt = dT(nt-1);
        K(:,1) = f(t,xt);
        for nstage=2:Nstage
            a=A(nstage,1:nstage-1)';
            K(:,nstage) = f(t+dt*c(nstage),xt+dt*(K(:,1:nstage-1)*a));
        end
        xt = xt + dt*(K*b);
        x(:,nt) = xt;
    end
end
```

By using this function, one can easily simulate a model with several ERK methods in an organized and effective way.

Figure 1 shows the true solution and the simulation results for each of the considered ERK methods.

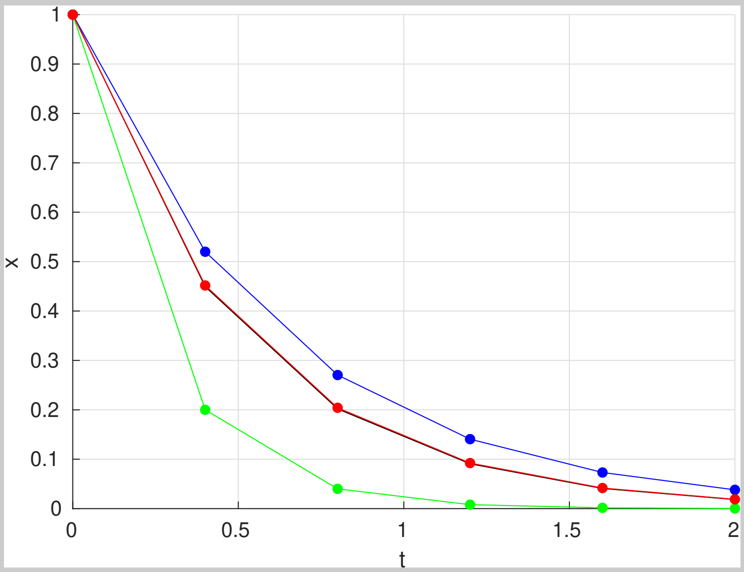

Figure 1: True solution (black) and simulation results for RK1 (green), RK2 (blue) and RK4 (red) as a function of t_k.

(b) Investigate the evolution of the accuracy of the integrators (against the true solution of (7)) as a function of the time-step Δt. Moreover, find the actual order of the methods as a function of the time-step.

Add the corresponding plots to your answer, and compare the obtained results to the theoretical order of accuracy of the various schemes.

Solution: Figure 2 shows the largest local approximation error for a given time-step.

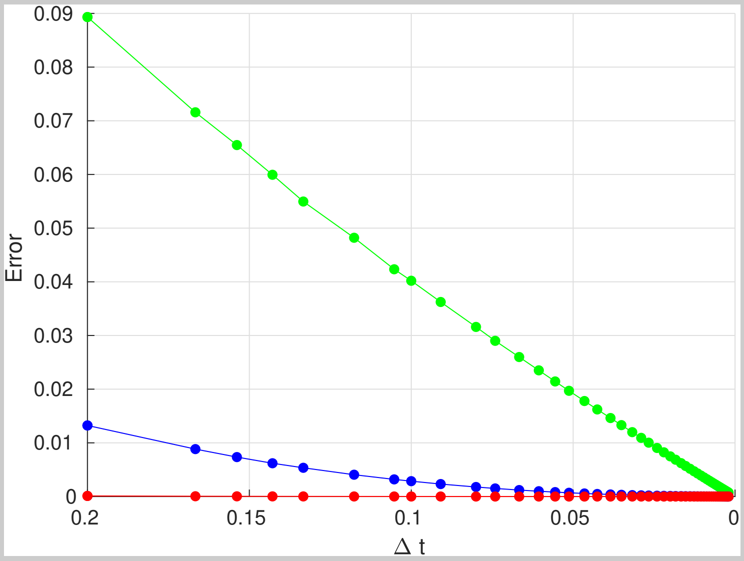

Figure 2: Largest local error for RK1 (green), RK2 (blue) and RK4 (red) as a function of Δt_k.

The order of a RK method is calculated based on the global approximation error. Figure 3 shows the order of each ERK method as a function of the time-step. As we can see, the orders of RK1, RK2 and RK4 are the theoretical ones for small time-steps. This is as expected.

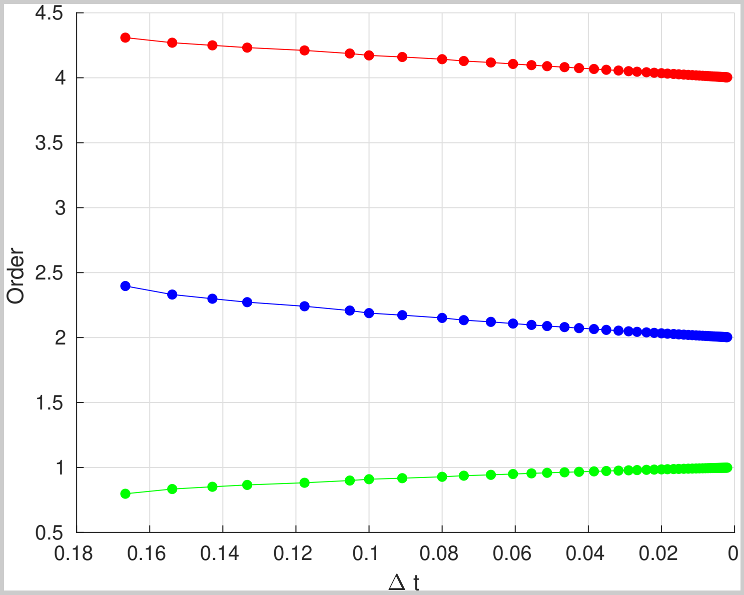

Figure 3: Order for RK1 (green), RK2 (blue) and RK4 (red) as a function of Δt_k.

(c) For what value of $\lambda < 0$ will the different schemes become unstable?

Solution: The schemes are stable if and only if $|R(\lambda\Delta t)| \leq 1$, where R is the corresponding stability function.
For RK1, RK2 and RK4, this condition is respectively equivalent to

$$\begin{aligned} |1+\lambda\Delta t| \leq 1 \quad &\Leftrightarrow \quad \lambda\Delta t \in [-2,0]\,, \\ \left|1+\lambda\Delta t+\frac{1}{2}(\Delta t)^2\right| \leq 1 \quad &\Leftrightarrow \quad \lambda\Delta t \in [-2,0]\,, \\ \left|1+\lambda\Delta t+\frac{1}{2}(\Delta t)^2+\frac{1}{6}(\Delta t)^3+\frac{1}{24}(\Delta t)^4\right| \leq 1 \quad &\Leftrightarrow \quad \lambda\Delta t \in [-2.7853,0] \quad \text{(Approximated value).} \end{aligned}$$

Hence, the RK1 and RK2 schemes will be unstable for $\lambda < -\frac{2}{\Delta t}$, while the RK4 scheme will be unstable for $\lambda < -\frac{2.7853}{\Delta t}$.

Problem 3 (Van der Pol oscillator)

Consider the nonlinear dynamics:

$$\dot{x} = y \tag{8a}$$
$$\dot{y} = u\left(1-x^2\right)y - x\,, \tag{8b}$$

with $u = 5$ and initial conditions $x(0) = 2$ and $y(0) = 0$.

(a) Simulate the dynamics (8) using the adaptive integrator `ode45` for a final time $t_f = 25$, and with the default solver options. Plot the obtained results and the discrete times selected by `ode45`. What do you observe? Explain.

Solution:

Figure 4 shows $y(t)$ versus $x(t)$, Figure 5 shows $x(t)$ and Figure 6 shows $y(t)$, while Figure 7 shows the time-steps provided by `ode45`.

We observe that $x(t)$ and $y(t)$ are periodic functions with the same period. These functions are oscillatory in a non-linear way. Moreover, the time-steps provided by `ode45` decrease in the regions, where the states vary the most, and vice versa. This is as expected from an adaptive integrator.

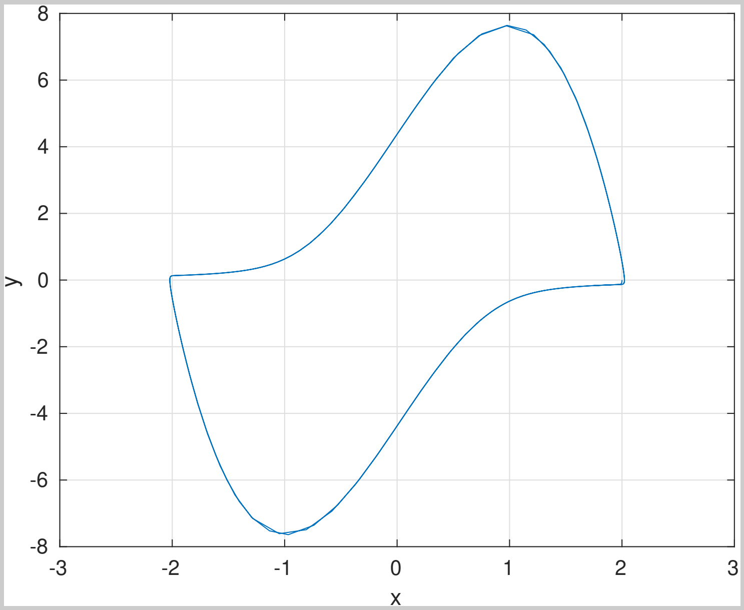

Figure 4: $y(t)$ vs. $x(t)$ provided by `ode45`.

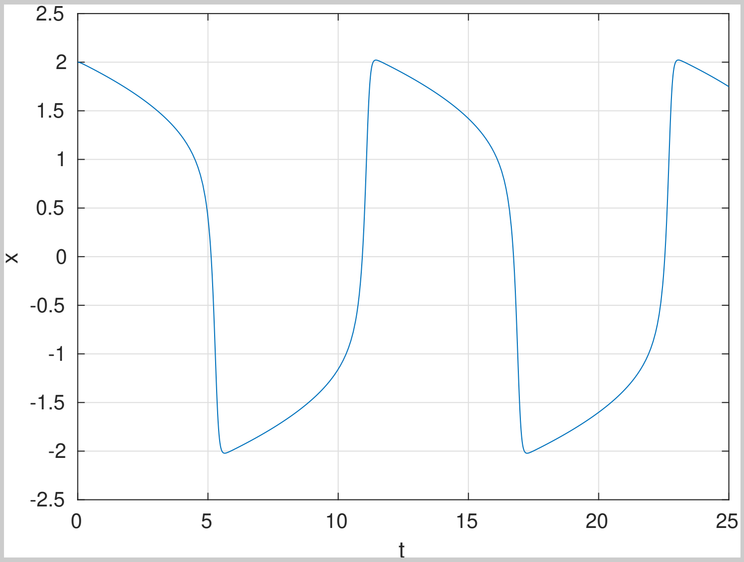

Figure 5: $x(t)$ provided by `ode45`.

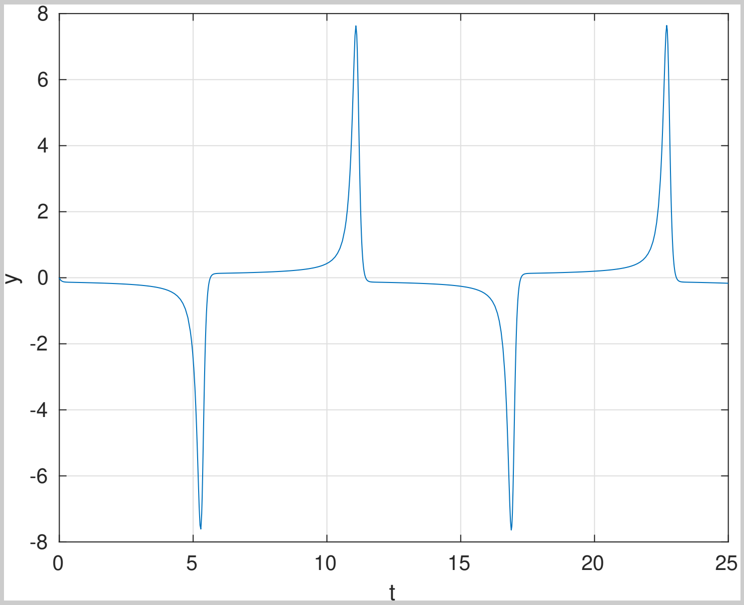

Figure 6: $y(t)$ provided by `ode45`.

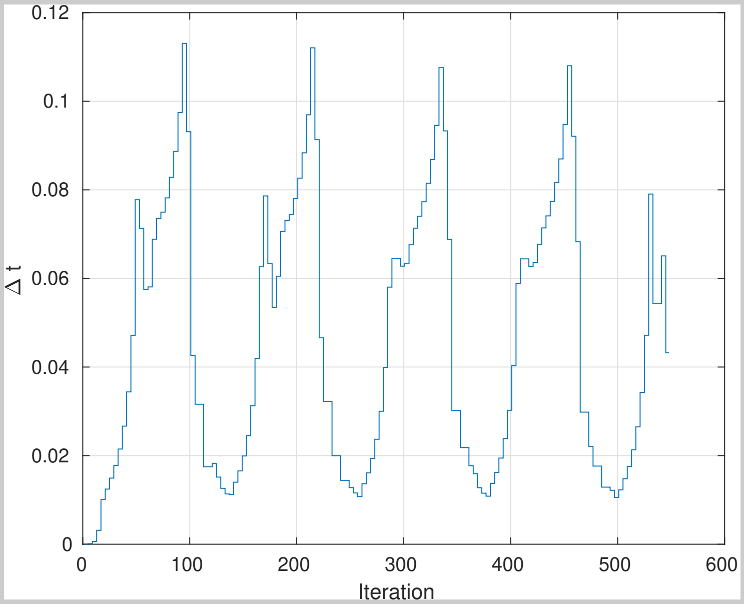

Figure 7: Δt_k provided by `ode45`.

(b) Simulate the dynamics (8) with your own RK4 scheme using $\Delta t = 0.1$, and compare the results. What time-step size is necessary so that the results from RK4 match the ones from `ode45`? Compare your discrete time grid to the one of `ode45`.

Solution: Figure 8 compares $[x(t), y(t)]$ for the 2 solvers, while Figure 9 and Figure 10 compare $x(t)$ and $y(t)$, respectively.

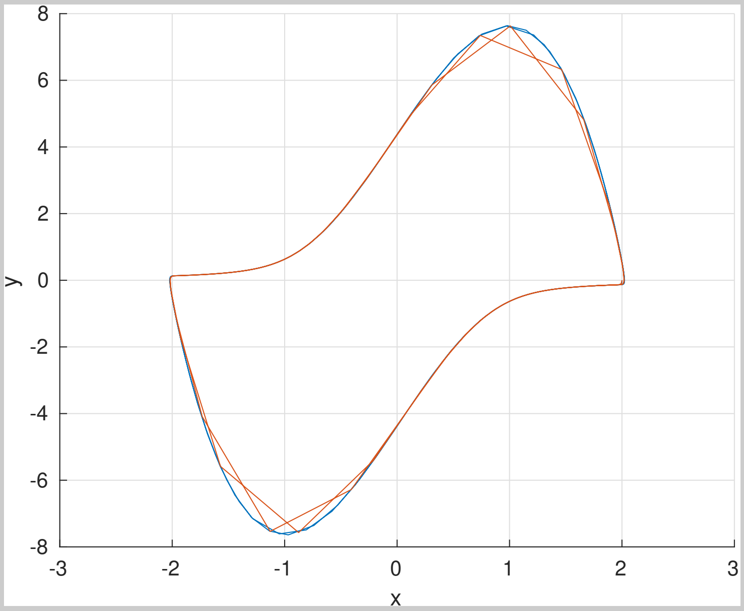

Figure 8: $y(t)$ vs. $x(t)$ provided by `ode45` (blue) and RK4 (red).

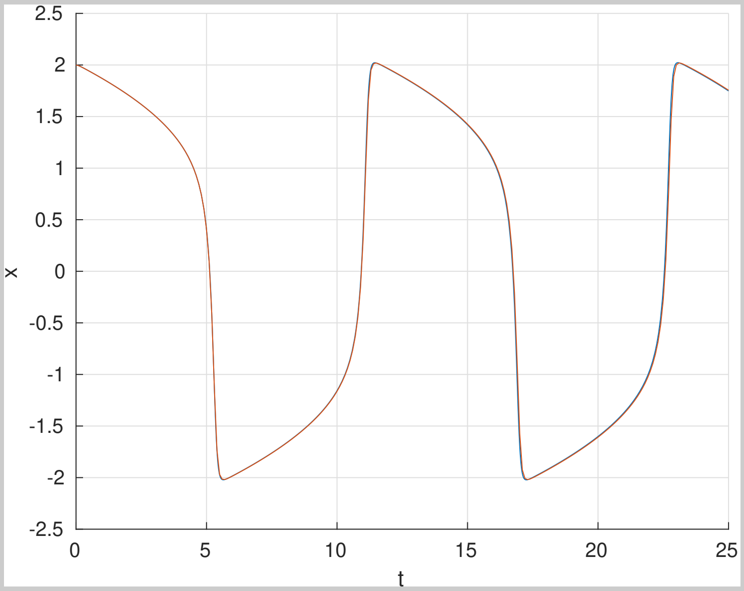

Figure 9: $x(t)$ provided by `ode45` (blue) and RK4 (red).

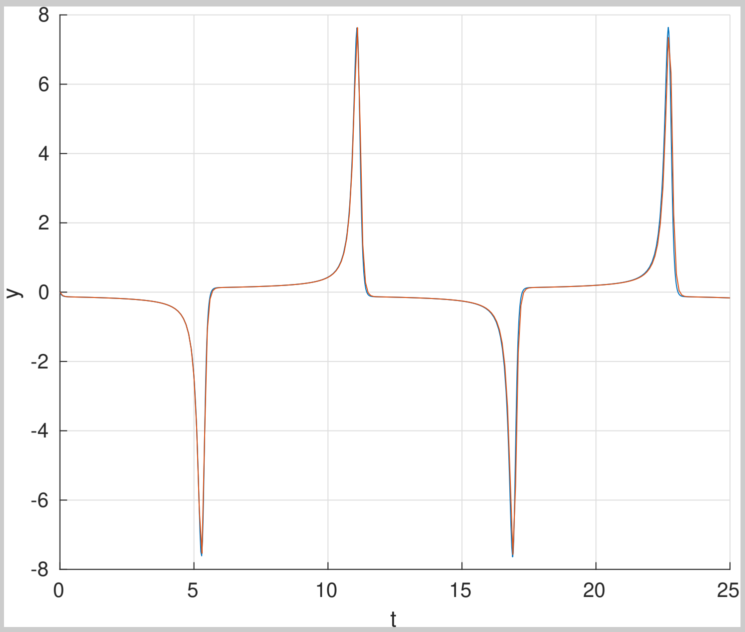

Figure 10: $y(t)$ provided by `ode45` (blue) and RK4 (red).

The time-step required to accurately approximate the solution from `ode45` is around 0.01 (or less). For this time-step the RK4 method requires 2500 iterations, while the `ode45` only requires 548 steps. In other words, for the same accuracy, `ode45` requires almost 5 times less iterations than the RK4 method. This is why adaptive methods are in general more computational efficient than ERK methods.