ESS101 Mod & Sim

Assignment 3 - Explicit and implicit integrators

Due date:
22/10/2025, 23:59

In this assignment we will investigate some widely-used explicit and implicit integration techniques, and their behaviour for the simulation of various dynamic systems.
NB. Note that there is an oral examination of this assignment, see instructions below!

Instructions

The assignments comprise an important part of the examination in this course. Hence, it is important to comply with the following rules and instructions:

- The assignment is pursued in groups of two students. For group formation procedures, see Canvas.
- For this assignment, there will be an **oral examination**. Each group meets with a TA and should be ready to answer questions on their findings from the assignment. **Please bring figures and results from simulations to support your answers**! There will also be questions on the Matlab code submitted, to check that each student has an understanding of what is going on in the code.
- In addition to the oral examination, each group should write a **one-page "executive summary"** on the tasks carried out. Include only brief comments and no figures!
- The one-page "executive summary" should be uploaded to Canvas *before the deadline*. Matlab code is uploaded as separate files. If the deadline is not met, we cannot guarantee that you will be able to complete the assignment during the course.
- The oral examination is graded PASS or FAIL.
- Since the assignments are part of the examination in the course, plagiarism is of course not allowed. If we observe that this happens anyway, it will be reported.

First part – explicit integration schemes

1. **General Runge-Kutta 2 methods** For a dynamic model:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}, t) \tag{1}$$

where for the sake of simplicity we assume the input $\mathrm{u} = \mathrm{u}_k$ to be constant on the time interval $[t_k, t_{k+1}]$, RK2 methods are based on 2 evaluations of the function f, the first of which occurs at $\mathrm{x}(t_k), \mathrm{u}_k$. They can generally be written as:

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

and have the Butcher tableau (zeros in the "a" part are traditionally omitted)

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

The RK2 method will have a numerical error $e_k = \mathrm{x}_{k+1} - \mathrm{x}(t_{k+1})$ (where $\mathrm{x}(t)$ is the true trajectory of the ODE).

(a) For the case of simplicity, consider the case where f is time-independent (t does not enter as an argument) and $\mathrm{u}(t) = \mathrm{u}_k$ is constant over the time interval $[t_k, t_{k+1}]$. Provide conditions on a, b such that the error e_k of the method is of order 3.
Hint: observe that:

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

(b) RK2 methods are at best of order 2. How does that relate to $e_k = \mathscr{O}\left(\Delta t^3\right)$?

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

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

For what values of n, b, c, the RK2 method is exact on (4)? What do you observe from your computations?

2. **Accuracy & stability** We will start with investigating the accuracy/computational cost of some explicit integration schemes. We will consider the explicit Euler scheme and a Runge-Kutta scheme of order 2 and 4. We will test them on the classic test system:

$$\dot{x} = \lambda x \tag{14}$$

where $\lambda < 0$. The Butcher tables of these three schemes are provided below.

(a) Code an explicit Euler scheme (RK1), and an RK2 and RK4 scheme for a generic function $f(x)$.
Test your codes on the system (14) with $\lambda = -2$ and $\Delta t = 10^{-1}$, simulating for $t_\mathrm{f} = 2$. Start with the initial condition $x(0) = 1$.
Hint: Declare your dynamics in a symbolic way, and generate a Matlab function that evaluates it, using "matlabFunction". The most comfortable way of coding this is also to have a code that "reads" the Butcher Tableau and deploys from it the adequate steps.

(b) Plot the accuracy (i.e. the error relative to the true solution of (14)) of your integrators vs time step Δt. Compare your results with the theoretical order of accuracy of the various schemes.
Hint: a plot similar to Figure 6.5 in the Lecture Notes is useful.

(c) For what value of $\lambda < 0$ will the different schemes become unstable? Use the same parameters as in (a) (except λ).
Hint: it is fine to check this by simulations with λ real.

Table 1: Explicit Euler

0	
	1

Table 2: Runge-Kutta 2

0		
$\frac{1}{2}$	$\frac{1}{2}$	
	0	1

Table 3: Runge-Kutta 4

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$		$\frac{1}{2}$		
1			1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

3. **Van-der-Pol oscillator** Consider the nonlinear dynamics:

$$\dot{x} = y \tag{15a}$$

$$\dot{y} = u\left(1 - x^2\right) y - x \tag{15b}$$

for $u = 5$, initial condition $(x(0), y(0)) = (1, 0)$, a final time of $t_\mathrm{f} = 25$.

(a) Simulate the dynamics (15) using the Matlab integrator ode45 with default options (ode45 is an adaptive Runge-Kutta scheme of 5 with alternate Butcher tableau of order 4). Plot the discrete times selected/reported by ode45. What do you observe?
Hint: to set the default options for ode45*, use the command* `options = odeset();`

(b) Deploy your own RK4 scheme on the same dynamics, and compare the solutions. What Δt do you need in order for your solution to accurately match the one of the function ode45, now using tighter tolerances equal to 10^{-8}? Compare your discrete time grid to the one of ode45.
Hint: to set the tighter tolerances for ode45*, use the command*
`options = odeset('AbsTol',1e-8,'RelTol',1e-8);`

Second part – implicit integration schemes

4. **IRK** We will now use an IRK scheme with $s = 2$ stages and of order $2s = 4$, having the Butcher Tableau provided below.

Table 4: IRK4

$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

(a) Write a Matlab code that deploys the IRK scheme on dynamics of the form:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, t) \tag{16}$$

i.e. we will omit inputs for the sake of simplicity.
Note that there is no question to answer here, except the code itself.
Hint: this task can be fairly simple. Start with writing a code that formulates $\mathrm{r}(\mathrm{K}, \mathrm{x}_k)$ *from symbolic data* A, b, c *(where A is a matrix, and b and c are vectors). Then export a Matlab function that evaluates* r *and* $\frac{\partial \mathrm{r}}{\partial \mathrm{K}}$*. In order to make your life easy, it is arguably best to declare your* $\mathrm{K}_{1,\ldots,s}$ *as a matrix (in* $\mathbb{R}^{n\times s}$*), and use the "reshape" command to cast them as a vector. Your main code will then simply have a Newton iteration (to solve* $\mathrm{r} = 0$ *for* $\mathrm{K}_{1,\ldots,s}$ *at each integrator step) nested in a for loop that runs the simulation.*

(b) Test your code vs. the explicit RK4 scheme you have developed in question (2) on the test system

$$\dot{x} = \lambda x, \tag{17}$$

using $\Delta t = 0.1$, $t_f = 1$, and $x(0) = 1$. Investigate the stability for the two schemes with different λ.

(c) Deploy your IRK4 and explicit RK4 on the Van Der Pol dynamics:

$$\dot{x} = y \tag{18a}$$

$$\dot{y} = u\left(1 - x^2\right) y - x \tag{18b}$$

for $u = 5$, initial condition $(x(0), y(0)) = (1, 0)$, a final time of $t_\mathrm{f} = 25$ and $\Delta t = 10^{-2}$. Compare your solutions.