Technical University of Munich
School of Computation, Information and Technology
Vikas Kurapati
Chinmay Datar
Ana Cukarska
Felix Sievers

WT 2025/2026

Lab Course Scientific Computing

Worksheet 3

distributed: Thu., 04.12.2025
due: Sun., 14.12.2025, 23:59 (submission on the Moodle page)
oral examination: Tue., 16.12.2025 (exact time slots announced on the Moodle page)

We now reconsider the two ODEs from worksheet 2 but with different parameters

For **Dahlquist's test equation** we now consider $\lambda = -7$ such that we get:

$$\dot{x} = (-7)x \tag{1}$$

with initial condition

$$x(0) = 1. \tag{2}$$

The analytical solution is then given by

$$x(t) = e^{-7t}.$$

a) Reuse the Euler method implemented in worksheet 2 to compute approximate solutions for equation (1) with initial condition (2), end time $t_{end} = 5$, and $\delta t = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$.

Plot your solutions for $t \in [0, 5]$ in one figure together with the analytical solution. Plot only in the range $x \in [-1, 1]$.

b) We will now implement the implicit Euler method. As a first step, implement the Newton method to solve a nonlinear equation $\mathcal{G}(a) = 0$. The update step for the Newton iteration reads:

$$a^{n+1} = a^n - \mathrm{D}\mathcal{G}(a^n)^{-1}\mathcal{G}(a^n). \tag{3}$$

Calculate the derivative analytically and pass the function handle as an argument to the Newton solver, i.e.: `function y = newton(x_0, G, dG)`.
As stopping criteria for the method use a maximum number of iterations of 100 and accuracy of $\epsilon = 10^{-8}$.

Hint: To make sure your implementation works properly, you may test it with this problem: find the root of $x^3 - 3$. Start at $x_0 = 1$.

c) Use the Newton method to implement the implicit Euler method with variable step size δt and end time t_{end} for the solution of the initial value problem

$$\dot{y} = f(y), \quad y(0) = y_0$$

as a function of the initial value y_0, the time step size δt, the end time t_{end}, the right hand side $f(y)$, and the first derivative of the right hand side with respect to y, i.e.:
`function y = impl_euler(y_0, dt, t_end, f, df)`
The output of the function is a vector containing all approximate values for y, including y_0.

Hints:

- The update scheme of the implicit Euler reads $y^{n+1} = y^n + \delta t \cdot f(y^{n+1})$. How do you map this to a root finding problem?
- Examine if the equation to be solved in each time step is solvable. If not, stop the time stepping with the method and the time step concerned and do not consider the associated approximations of y in your further examinations.

d) Compute – as far as possible – approximate solutions for equation (1) with initial conditions (2) and with time steps $\delta t = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$. Plot your solutions for $t \in [0, 5]$ in one graph together with the analytical solution. Plot only in the range $x \in [-1, 1]$.

e) To compare the results of the implicit method to those of the explicit method, we compute again the approximation error

$$E = \sqrt{\frac{\delta t}{t_{end}} \sum_k \left(x_k - x_{k,exact}\right)^2}$$

for each case, where x_k denotes the approximation of $x(\delta t \cdot k)$ and $x_{exact,k}$ the exact values of x at $t = \delta t \cdot k$. Additionally, determine the factor by which the error is reduced if the step size δt is halved.

Collect the results in the tables below. Also write all information needed in the tables to the MATLAB console in a readable way.

Hint: The reduction factor should be greater than one if the error is reduced.

f) In addition to accuracy, we examine an additional aspect of 'quality' of a method: the *stability*. Descriptively spoken, stability denotes the applicability of a method for varying parameters, whereas at least results similar to the exact/correct solution have to be achieved (In particular, unphysical oscillations should not occur). With this heuristic definition, decide for which of the used values for δt each of the examined methods is stable (in the case of our problem).

Mark stable cases by a cross in the last table. Try to find a simple criterion to determine whether a solution is stable or not and write the result to the MATLAB console as well.

Hint: It is **not** necessary to perform a von Neumann stability analysis here!

explicit Euler

δt	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$
error					
error red.	—				

implicit Euler

δt	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$
error					
error red.	—				

Stable cases

δt	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$
Explicit Euler					
Implicit Euler					

Now for the **Van-der-Pol-Oscillator** we consider $\mu = 4$:

$$\begin{cases} \dot{u} = v \\ \dot{v} = 4(1-u^2)v - u \end{cases} \tag{4}$$

with initial conditions

$$\begin{aligned} u(0) &= 1, \\ v(0) &= 1. \end{aligned} \tag{5}$$

g) Try to solve the equation with the explicit Euler method. Use $t_{end} = 20$ and $\delta t = 0.1$. Note that you will have to extend your implementation to account for vector-valued functions. What do you observe?

h) Now extend your Newton Solver to vector-valued problems as well. Since the derivative is now a matrix (the Jacobian), use MATLAB's builtin '\' operator to solve the linear system.

Hint: Take care to replace 1 by the identity matrix where necessary.

i) Solve the Van-der-Pol-Oscillator with the implicit Euler method using the extended Newton solver. Note that you will have to extend the implicit method implementation as well in order to handle vector-valued functions. Use the same parameters as in **g)**. Plot your solutions as in the previous worksheet: u vs. t, v vs. t, and v vs. u as subplots in the same figure.

Hint: The Jacobian of the right-hand side is:

$$\mathrm{D}f(u,v) = \begin{pmatrix} 0 & 1 \\ -2\mu uv - 1 & \mu(1-u^2) \end{pmatrix}$$

j) Now set the time step to $\delta t = 1.0$ using the implicit Euler scheme (all other parameters from **g)**). If the Newton solver fails to converge (**hint:** this is expected here), terminate the integration and display an informative error message, such as:

- *Newton solver exceeds maximum iterations.*
- *Singular matrix in Newton solver.*

Questions:

Q1 For which integer q can you conclude that the accuracy of the

a) explicit Euler method,

b) implicit Euler method,

behaves like $O\left(\delta t^q\right)$?

Q2 In the lecture, we learned the implicit Euler method is *unconditionally* stable, ensuring stable solutions for any δt. However, in this worksheet, we mentioned and also observed that the integration scheme occasionally becomes unsolvable for specific δt values. Can you explain this discrepancy? What are the common reasons for the scheme (in our case implicit Euler and Newton) being unable to find a solution?

Q3 Which type of methods (explicit/implicit) would you choose for problems such as those from this worksheet (Dahlquist's equation and the Van-der-Pol-Oscillator)? Give a reason why you would choose a certain type of method and not the other.

Q4 Can you give a real world example, where you need an explicit time stepping scheme? Can you give a real world example, where you need an implicit time stepping scheme?

Q5 Why do we need the built-in backslash operator in the vector-valued Newton solver in **h)**?