Dr. Markus Faustmann
Paul Dunhofer
Institute of Analysis and Scientific Computing
Summer term 2025

due on 21.03.2025

# Scientific programming for interdisciplinary mathematics - Worksheet 2

*Topics: numerical methods for nonlinear ODEs (fixed-point, Newton method)*

---

### Fixed-point iteration

Given a nonlinear function $G : \mathbb{R}^d \to \mathbb{R}^d$, a *fixed-point problem* seeks $z^\star \in \mathbb{R}^d$ such that

$$G(z^\star) = z^\star. \tag{1}$$

Suppose that $G$ is a *contraction*, i.e., with Lipschitz constant $0 < q < 1$ it holds that

$$\|G(y) - G(z)\| \leq q\,\|y - z\| \quad \text{for all } y, z \in \mathbb{R}^d,$$

where $\|\cdot\|$ is an appropriate norm on $\mathbb{R}^d$. Given an initial iterate $z_0 \in \mathbb{R}^d$, the successive application of $G$ defines the sequence $z_{j+1} := G(z_j)$ for $j \in \mathbb{N}_0$. The Banach fixed-point theorem proves that $(z_j)_{j\in\mathbb{N}_0}$ converges linearly towards the unique fixed point $z^\star \in \mathbb{R}^d$, i.e.,

$$\|z^\star - z_j\| \leq q^{j-k}\,\|z^\star - z_k\| \quad \text{for all } j, k \in \mathbb{N}_0.$$

### Stopping criteria for iterative methods

Given parameters $\tau_{\text{abs}} > 0$ and $J_{\max} \in \mathbb{N}$, the following criteria are employed to decide the acceptance of an iterate $z_j$ or the break-off of an iterative method:

- *Absolute error tolerance* $\|z_{j+1} - z_j\| \leq \tau_{\text{abs}}$ (`abs`)
- *Maximal number of iterations* $J_{\max} \leq j$. (`iter`)

In practice, an iteration is considered as converged if the condition (`abs`) is satisfied. In this case, the final iterate $z_{j+1}$ is the desired approximation of the fixed-point $z^*$. If instead the maximum number of iterations is reached (`iter`) before the other criteria are satisfied, the iteration has not converged. The choice an appropriate parameter $\tau_{\text{abs}} > 0$ clearly has an impact on the computational cost. For the exercises below, we use

$$\tau_{\text{abs}} = 10^{-12} \quad \text{and} \quad J_{\max} = 1000.$$

**Problem 1:**
Implement a MATLAB function for the fixed-point iteration. Adhere to the signature

$$[\texttt{z}, \texttt{zvec}] = \texttt{fixedpoint}(\texttt{G}, \texttt{z0})$$

The arguments consist of the function handle `G` for the contraction $G : \mathbb{R}^d \to \mathbb{R}^d$ and the initial iterate `z0`. The output $\texttt{z} = z_{j+1}$ is the approximation of the fixed point $z^\star$, where the stopping condition (`abs`) is satisfied with $\tau_{\text{abs}} = 10^{-12}$. Use $J_{\max} = 1000$ and the Euclidean norm on $\mathbb{R}^d$ (which is `norm` in MATLAB). The columns $z_\ell = \texttt{zvec}(:, \ell + 1)$ of $\texttt{zvec} \in \mathbb{R}^{d\times(j+1)}$ should be the fixpoint iterates.

**Problem 2:**

For any parameter $0 < \alpha < 1$, the function $G : \mathbb{R} \to \mathbb{R}$, $G(z) := \exp(-\alpha|z|)$ is a contraction. Apply your implementation of the fixed-point iteration from Exercise 1 with initial guess $z_0 = 100$ to approximate the fixed point $z^\star$ with $G(z^\star) = z^\star$.

**Problem 3:**

Given a right-hand side $f : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ and an initial value $y_0 \in \mathbb{R}^d$, consider a (possibly nonlinear) initial value problem

$$y'(t) = f(t, y(t)) \quad \text{for } t \in [0, T] \quad \text{and} \quad y(0) = y_0. \tag{2}$$

Implement the following *one-step methods* for the solution of the initial value problem (2), where the function parameters are as follows: $\texttt{t} = (t_0, \ldots, t_L) \in \mathbb{R}^{L+1}$ is the row vector of the time steps and $\texttt{y0} = y_0 \in \mathbb{R}^d$ is the initial condition as a column vector and `f` is a function handle for the $f$-function. The return value should be the matrix $\texttt{y} = (y_0, \ldots, y_L) \in \mathbb{R}^{d\times(L+1)}$, where the columns $y_\ell = \texttt{y}(:, \ell) \in \mathbb{R}^d$ are the approximations $y_\ell \approx y(t_\ell)$.

(a) Explicit Euler method $\quad \texttt{y} = \texttt{explicitEuler}(\texttt{t}, \texttt{f}, \texttt{y0})$

(b) Implicit Euler method $\quad [\texttt{y}, \texttt{iter}] = \texttt{implicitEuler}(\texttt{t}, \texttt{f}, \texttt{y0})$

This requires the solution $z = y_{j+1}$ of the fixed-point problem $z = G(z) := y_j + (t_{j+1} - t_j)\, f(t_{j+1}, z)$ with your function `fixedpoint` from Exercise 1. Use $z_0 = y_j$ as initial iterate of the fixed point iteration. Provided that $(t_{j+1} - t_j)$ is sufficiently small, one can usually show that this equation admits a unique solution. In addition to `y`, also return the row vector $\texttt{iter} \in \mathbb{N}^L$, which contains the number of iterations of `fixedpoint` for each time step.

(c) Implicit midpoint method $\quad [\texttt{y}, \texttt{iter}] = \texttt{implicitMidpoint}(\texttt{t}, \texttt{f}, \texttt{y0})$

Proceed analogously to part (b) for the solution of the fixed-point problem iteration of the one-step method, where we approximate the fixed point $z = y_{j+1}$ of the problem

$$z = G(z) := y_j + (t_{j+1} - t_j)\, f\Big(\frac{t_{j+1} + t_j}{2}, \frac{y_j + z}{2}\Big)$$

with initial value $z_0 = y_j$.

**Problem 4:**

The task is to investigate the number $y$ of individuals infected by a nondeadly disease with permanent immunity in a population. (Consider $y$ as a continuous number, e.g., the percentage.) The growth of the quantity $y$ depends on the growth factor $k > 0$ and the saturation $C - y$ for the maximal capacity $C > 0$. Let $y_0 \geq 0$ denote the initial number of infections at time $t = 0$. This leads to the *nonlinear* initial value problem of the *logistic equation*

$$y'(t) = k\, y(t)\, (C - y(t)) =: f(t, y(t)) \quad \text{with initial value} \quad y(0) = y_0. \tag{3}$$

For simplicity, set $k = C = 1$ and $y_0 = 1/2$. The exact solution reads $y(t) := \big(1+(y_0^{-1}-1)\exp(-t)\big)^{-1}$. Apply the functions from Exercise 3 to approximate the solution to the initial value problem (3) for $t \in [0,5]$.

(a) Plot the exact and discrete solutions into a single plot.

(b) Plot the number of fixed-point iterations of `implicitEuler` and `implicitMidpoint` (i.e., `iter`) in a separate plot.

(c) Compute the exact error $e_h := \max_{\ell=1,\dots,L} |y(t_\ell) - y_\ell|$ in the maximum norm in dependence of the uniform step sizes $h \in \{2^{-n} \,:\, n = 1,\dots,N\}$ and compare the convergence rates for the different methods.

**Newton's method**

The *Newton method* is used to numerically compute a zero $z^\star \in \mathbb{R}^d$ of a given function $F\colon \mathbb{R}^d \to \mathbb{R}^d$, i.e., $F(z^\star) = 0$. For an initial value $z_0 \in \mathbb{R}^d$, we define the sequence of approximations $(z_j)_{j\in\mathbb{N}_0} \approx z^\star$ inductively via

$$z_{j+1} := z_j + \delta_j, \quad \text{where} \quad \delta_j = -\mathrm{D}F(z_j)^{-1}\,F(z_j) \quad \text{for all } j \in \mathbb{N}_0, \tag{4}$$

with the Jacobian $\mathrm{D}F(z_j) \in \mathbb{R}^{d\times d}$. Note that any fixed-point problem $G(z^\star) = z^\star$ with $G\colon \mathbb{R}^d \to \mathbb{R}^d$ and sought fixed point $z^\star$ can be recast as a zero problem $F(z^\star) = 0$ with the residual $F(z) := G(z) - z$.

**Problem 5:**

Implement Newton's method as a MATLAB function of the form

$$[\texttt{z}, \texttt{zvec}] = \texttt{newton}(\texttt{F}, \texttt{DF}, \texttt{z0})$$

Here, `F` is a function handle for the objective function $F$, `DF` is a function handle for its Jacobian, and the column vector $\texttt{z0} = z_0 \in \mathbb{R}^d$ is the initial iterate. The column vector $\texttt{z} = z_j \in \mathbb{R}^d$ is the final iterate, where the stopping criterion `abs` is satisfied with $\tau_{\text{abs}} = 10^{-12}$ and $J_{\max} = 1000$. The columns $z_\ell = \texttt{zvec}(:, \ell + 1)$ of $\texttt{zvec} \in \mathbb{R}^{d\times(j+1)}$ should be the Newton iterates.

**Problem 6:**

Test your function `newton` from Exercise 5 using $z_0 = 5$ with the following scalar examples:

(a) $F(z) = z^2 + \exp(z) - 2$ in $[0, 10]$ (reference solution $z^\star = 0.5372744491738566$)

(b) $F(z) = \log(z) + z^2$ in $[1/4, 10]$ (reference solution $z^\star = 0.6529186404192047$)

(c) $F(z) = (\cos(2z))^2 - z^2$ in $[1/4, 10]$ (reference solution $z^\star = 0.5149332646611294$)

If the iteration converges, compute the error $e_k := |z_k - z_{k-1}|$ of two consecutive iterates and the convergence order

$$q_k := \log(e_k/e_{k-1})/\log(e_{k-1}/e_{k-2}).$$

Plot the convergence order $q_k$ in each iteration step, i.e., plot $q_k$ (on the $y$-axis) over the iterations $k$ (on the $x$-axis).

**Problem 7:**
The *Lotka–Volterra equation* provides a simple model for the population of two species $y_1$ and $y_2$ such that species $y_1$ has unlimited food and can grow unlimitedly, whereas species $y_2$ eats the species $y_1$. The corresponding initial value problem seeks $y \in C^2([0,T];\mathbb{R}^2)$ such that

$$y'(t) = f(y(t)) \quad \text{with} \quad y(0) = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} \tag{5}$$

with the nonlinear function $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$f(y) \coloneqq \begin{pmatrix} y_1(1-y_2) \\ -y_2(1-y_1) \end{pmatrix}.$$

(a) Solve the initial value problem (5) for $T = 15$ using the one-step methods from Exercise 3.

(b) In the implicit methods in part (a), replace the fixed-point iteration with your implementation `newton` of Newton's method. Save the adapted functions as `implicitEulerNewton` and `implicitMidpointNewton`. Compare the number of necessary iterations per time-step.

**Adaptive time-step control**

In the first two weeks of this course, we have looked at explicit and implicit one-step methods to solve initial value problems. In practice, one is not interested in methods with a constant step size. Instead, sophisticated computer programs for the numerical solution of initial value problems work with adaptive strategies, in which the computer selects an optimal step size $h_\ell$ in each step in order to achieve a given accuracy $\tau > 0$ as efficiently (i.e., with as few evaluations of $f$) as possible. The following pseudocode illustrates the theoretical procedure algorithmically.

**Input:** time interval $[t_0, T]$, initial value $y_0$, right-hand side $f$ of the initial value problem $y(t_0) = y_0$ and $y'(t) = f(t,y)$ for $t \in [t_0, T]$, one-step method with incremental function $\Phi$, tolerance $\tau > 0$, counter $\ell = 0$.
**Goal**: Determine $t_0 < t_1 < \ldots < t_L = T$ and $y_\ell$ such that $\max_{\ell=1,\ldots,L} \|y(t_\ell) - y_\ell\|_\infty \lesssim \tau$.

`REPEAT`

- Determine $h_\ell > 0$ such that $\|z(t_\ell + h_\ell) - z_{\ell+1}\|_\infty \approx \tau\, h_\ell$, where $z_{\ell+1} \coloneqq y_\ell + h_\ell\, \Phi(t_\ell, y_\ell, h_\ell)$ and $z$ solves $z(t_\ell) = y_\ell$ and $z' = f(t,z)$ in $[t_\ell, T]$.
- Define $\quad t_{\ell+1} \coloneqq \min\{t_\ell + h_\ell, T\} \quad$ and $\quad y_{\ell+1} \coloneqq z_{\ell+1}$.
- Update counter $\ell \mapsto \ell + 1$.

`UNTIL` $\;t_\ell = T$
**Output:** mesh $\Delta = \{t_0, \ldots, t_L = T\}$, approximations $y_\ell \approx y(t_\ell)$ for all $\ell = 0, \ldots, L$.

**Simplified step size control based on $h$–$(h/2)$ strategy**

One method for adaptive step size control is the so-called $h$–$(h/2)$ strategy. In each step $\ell$, we solve the problem twice: first, one step of size $h$ yields $y_h \approx y_{\ell+1}$; second, two steps of size $h/2$ yield $y_{h/2} \approx y_{\ell+1}$. We then compare the error estimator $\varepsilon := \|y_h - y_{h/2}\|_\infty$ with a fixed *desired accuracy* $\tau > 0$:

- If $\varepsilon > \tau\, h$, we halve the step size $h \mapsto h/2$ and repeat the current step;
- If $\varepsilon \leq \tau\, h$, the desired accuracy was achieved and we move on with $y_{\ell+1} := y_{h/2}$ and:
  - If $\varepsilon \ll \tau\, h$, e.g., $\varepsilon < \tau\, h/2$, we double the current step size for the next step;
  - Otherwise, we keep the current step size $h$ also for the next step.

**Problem 8 (Implementation of the $h-(h/2)$ strategy):**
Implement a MATLAB function

$$[\mathtt{y}, \mathtt{t}] = \mathtt{adaptiveTimeStepping}(\mathtt{solveODE}, \mathtt{t0}, \mathtt{T}, \mathtt{f}, \mathtt{y0}, \mathtt{h0}, \mathtt{tau})$$

for the adaptive step size control based on the $h$–$(h/2)$ strategy. Here, the input `tau` is the desired accuracy $\tau > 0$, the column vector `y0` is the initial value $y_0 \in \mathbb{R}^d$, `t0` and `T` specify the time interval $[t_0, T]$, `h0` is the initial step size, and `f` is the function handle of the function $f\colon [t_0, T] \times \mathbb{R}^d \to \mathbb{R}^d$. Finally, `solveODE` is the function handle of a one step method with signature

$$\mathtt{z} = \mathtt{solveODE}(\mathtt{s}, \mathtt{f}, \mathtt{z0})$$

solving an ODE $z' = f(s, z)$ with initial value $z_0 = z(s_0) \in \mathbb{R}^d$ at the time steps $s_j$ specified in the row vector $\mathtt{s} = (s_0, \ldots, s_m) \in \mathbb{R}^{m+1}$. The output are the approximations $y_j \approx y(t_j) \in \mathbb{R}^d$ as a matrix $\mathtt{y} \in \mathbb{R}^{d\times(m+1)}$ at the corresponding adaptive time steps $\mathtt{t} = (t_0, \ldots, t_N) \in \mathbb{R}^{N+1}$ with $t_N = T$.

**Hint:** To obtain $y_h \approx y_{\ell+1}$, use the function `solveODE` with $s = [t_\ell, t_{\ell+1}]$ and $z_0 = y_\ell$. To obtain $y_{h/2} \approx y_{\ell+1}$, use the function `solveODE` with $s = [t_\ell, (t_\ell + t_{\ell+1})/2, t_{\ell+1}]$ and $z_0 = y_\ell$.

**Problem 9 (Solving an ODE system with adaptive time steps):**
In this exercise, we revisit the linear system of ordinary differential equations

$$\begin{Bmatrix} y_1' = -y_2 \\ y_2' = y_1 \end{Bmatrix} =: f(t, y) \quad \text{for } t \in [0, 2\pi] \quad \text{and} \quad \begin{Bmatrix} y_1(0) = 1 \\ y_2(0) = 0 \end{Bmatrix} \tag{6}$$

from Exercise 6 on Sheet 1. The exact solution reads $y(t) := (\cos(t), \sin(t))^\top$. For $n \in \mathbb{N}$, consider the explicit Euler method with *uniform mesh refinement* (i.e., uniform time steps) with $h := 2^{-n}$ and *adaptive mesh refinement* with $\tau := 2^{-n}$ and $h_0 := 5$. In either case, let $N \in \mathbb{N}$ be the number of time steps for each approach. Use your code from Exercise 8. Plot the errors after the final time step, i.e., $e_N := |y(2\pi) - y_N|$, on the y-axis over the number of time steps $N$ on the x-axis in a loglog-plot. What rate of convergence $e_N = \mathcal{O}(N^{-\alpha})$ do you observe?

**Take-home message**

For nonlinear ODEs, each step of an implicit Runge–Kutta method from Exercise sheet 1 requires the solution of a nonlinear system of equations. A simple way to solve the arising nonlinear system is by means of a fixed-point iteration which leads to linear convergence. To speed up the convergence, we can use the Newton method, which is a very efficient method for solving nonlinear systems and leads to quadratic convergence (at least locally in a vicinity of the solution).
In practice, adaptive time-step control is crucial for the efficient and accurate solution of initial value problems. The $h$–$(h/2)$ strategy is a simple and effective method to achieve this.