

# Programming Exam – Praktikum zur Programmierung in der Numerik

Numerical Analysis Group  
Eberhard-Karls-Universität Tübingen

02.03.2026 – 04.03.2026

## Exam Information

You have **three days** to complete the exam. The expected workload is approximately **8–12 hours** in total. Deadline is **Wednesday, 04.03.2026 at 17:00**. Late submissions will not be accepted.

Room **N14** in the **C-Bau** will be available for working on the exam. The administrator will be present there for questions and identity verification during the following hours:

- **09:00 – 12:00**
- **14:00 – 16:00**

You **MUST** verify your identity with the course coordinator by bringing your student ID when submitting the exam (submission is only possible from inside the university).

## Rules

- The only external Julia package you may use is **Plots**. Do *not* use **LinearAlgebra** or any other standard library beyond **Base**.
- All algorithms (norm computations, linear system solvers, decompositions, etc.) must be implemented by hand.
- Each exercise must be contained in a single **.jl** file that can be executed directly with **julia exerciseN\_name.jl**.
- The main section of each exercise must be wrapped in a **function main() ... end** and called at the end of the file. All top-level code (data definitions, function calls, plotting, printing) belongs inside **main()**.
- All plots must be saved as **.png** files using **savefig**. Include the generated image files in your submission.

## Academic Integrity

- The use of **AI-based tools** (e.g. ChatGPT, GitHub Copilot, Claude, Gemini, or any similar service) is **strictly forbidden**.
- **Collaboration** with other students is **not allowed**. This is an individual exam.
- **Copy-pasting code** from the internet, forums, or any other external source is **not allowed**. All code must be written by you.

All submissions will be checked for plagiarism and AI-generated content. Violations of any of the above rules **will lead to the failure of the exam**.

## Starter Files

Stub files for each exercise are provided in the `stubs/` folder included with this exam. Each stub contains the required function signatures, TODO comments, and the main section structure. You may use these as a starting point. **Remember to set the variable `d` to the last digit of your Matrikelnummer inside the `main()` function of each file.**

## Instructions for Exercises

Throughout this exam, let  $d$  denote the last digit of your Matrikelnummer (matriculation number). For example, if your Matrikelnummer is 4821376, then  $d = 6$ . Each exercise uses  $d$  to individualize certain parameters. Define  $d$  at the top of your `main()` function in each source file.

### Exercise 1: LR Decomposition

Write a single Julia file `exercise1_LR.jl` containing the following:

- a) A function `myLR(A)` that computes and returns the matrices  $L$  and  $R$  of the LR decomposition for a square matrix  $A \in \mathbb{R}^{n \times n}$ . You may assume that the input matrix  $A$  has a unique LR decomposition.
- b) A function `myLR_solve(L, R, b)` that, given lower triangular matrix  $L$ , upper triangular matrix  $R$ , and a vector  $b \in \mathbb{R}^n$ , computes and returns the solution  $x \in \mathbb{R}^n$  of the linear system  $(L \cdot R)x = b$  using forward and backward substitution.
- c) A main section that uses the functions from a) and b) to solve  $Ax = b$  and print the solution  $x$ . Use:

$$A = \begin{pmatrix} 4 & 0 & 0 \\ -2 & 6 & 2 \\ -4 & 3 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 16 + d \\ 8 + d \\ -20 + d \end{pmatrix}.$$

### Exercise 2: QR Decomposition

Write a single Julia file `exercise2_QR.jl` containing the following:

- a) A function `myQR(A)` that computes and returns matrices  $Q$  and  $R$  for a matrix  $A \in \mathbb{R}^{m \times n}$  with  $m \geq n$  and  $\text{rank}(A) = n$ , where  $Q \in \mathbb{R}^{m \times n}$  is the essential part of the orthogonal matrix and  $R \in \mathbb{R}^{n \times n}$  is an upper triangular matrix such that  $A = QR$ . Use the Householder algorithm.
- b) A function `myQR_solve(Q, R, b)` that computes the solution  $x \in \mathbb{R}^n$  of the linear system  $QRx = b$  by first computing  $Q^T b$  and then solving  $Rx = Q^T b$  using backward substitution.
- c) A main section that finds suitable coefficients  $\gamma_j$  for  $j = 1, \dots, 5$  so that the function  $f(x) = \sum_{j=1}^5 \gamma_j \phi_j(x)$  with basis functions

$$\phi_1(x) = x^2, \quad \phi_2(x) = x^4, \quad \phi_3(x) = \frac{1}{x^2}, \quad \phi_4(x) = \exp(-(x-1)), \quad \phi_5(x) = \sin(2\pi x)$$

is a least-squares fitting curve for the measurement data. Do not use the Gaussian normal equation.

The measurement data points are  $x_i \in \{0.1, 0.2, \dots, 1.0\}$  with corresponding values

$$f(x_i) = 100 + d, 14, 61, 68, 60, 35, 22, 80, 90, 105 + d.$$

Plot the fitting curve together with the measurement data. Use axis limits  $x \in [0, 1.1]$  and  $y \in [0, 140]$ .

## Exercise 3: Polynomial Interpolation

Let the function  $f : \mathbb{R} \rightarrow \mathbb{R}$  be defined by

$$f(x) = \frac{1}{1 + (25+d)x^2}.$$

Write a single Julia file `exercise3_interpolation.jl` containing the following:

- a) A function `NewtonInter(x, y, u)` that, for two input vectors  $x, y \in \mathbb{R}^{n+1}$  and a vector  $u \in \mathbb{R}^m$ , computes and returns the coefficients  $c$  of the Newton interpolation polynomial and the function values  $v_j = p(u_j)$  using divided differences. You may assume the interpolation points are distinct.
- b) A function implementing  $f(x)$ .
- c) A main section that interpolates  $f$  on the interval  $[-1, 1]$  with Chebyshev nodes  $x_i = \cos\left(\frac{(2i+1)\pi}{2n+2}\right)$  for  $i = 0, 1, \dots, n$  and  $n = 4, 8, 12$ . Plot the resulting polynomials together with  $f$  using axis limits  $x \in [-1.1, 1.1]$  and  $y \in [-0.5, 1.5]$ . Plot the interpolation error  $|p(x) - f(x)|$  for  $n = 12$  in a separate figure with  $x \in [-1.1, 1.1]$ .

## Exercise 4: Richardson Extrapolation

Write a single Julia file `exercise4_richardson.jl` containing the following:

- a) A function `richard(f, x0, h0, q, nmax, tol)` that approximates the derivative of a function  $f$  using Richardson extrapolation with central difference quotients. Use the Neville algorithm to build the extrapolation tableau with step sizes  $h_k = 2^{-k}h_0$  and extrapolation in powers of  $h^q$ . The procedure terminates when the error is  $< \text{tol}$  or  $nmax$  iterations are reached. Print the extrapolation tableau to the console and return the best approximation.
- b) A main section that approximates the derivative of  $f(x) = x^2 \sin(x)$  at  $x_0 = 1+d/10$  to at least 8 decimal places with  $h_0 = 1$  and  $q = 2$ , and prints the result.

## Exercise 5: Newton's Method

Write a single Julia file `exercise5_newton.jl` containing the following:

- a) A function `F(x)` implementing the 2D system of equations:  $x_1^2 + x_2^2 + \frac{3}{5}x_2 = \frac{4}{25}$  and  $x_1^2 - x_2^2 + x_1 - \frac{8}{5}x_2 = \frac{7}{50}$ .
- b) A function `NewtonV(F, x0, tol, maxiter)` that performs Newton's method until  $\|x_{k+1} - x_k\|_2 \leq \text{tol}$  or `maxiter` is exceeded. The function should return the iterate  $z$ , the number of iterations `iter`, and a flag  $b$  (success check). Approximate the Jacobian matrix using centered differences with  $h = 10^{-6}$ .
- c) A main section that tests Newton's method with initial values  $(1+0.1d, 1)^T$  and  $(-1-0.1d, -1)^T$  (`maxiter` = 100, `tol` =  $10^{-5}$ ) and prints the results.

## Exercise 6: Explicit Euler Method

The Van der Pol oscillator is described by the second-order ordinary differential equation

$$\ddot{z}(t) - \mu(1 - z(t)^2) \dot{z}(t) + z(t) = 0,$$

where  $\mu > 0$  is a parameter controlling the strength of the nonlinear damping.

Introducing  $z_1(t) = z(t)$  and  $z_2(t) = \dot{z}(t)$ , this equation transforms into the equivalent first-order system

$$z_1'(t) = z_2(t), \quad z_2'(t) = \mu(1 - z_1(t)^2) z_2(t) - z_1(t).$$

Write a single Julia file `exercise6_euler.jl` containing the following:

- a) A function `ExplEuler(a, b, N, start, mu)` that approximates this first-order system on the interval  $[a, b]$  with  $N$  equidistant subintervals using the explicit Euler method. Return the time vector and the solution matrix.
- b) A main section that:
  - (i) Computes a reference solution on  $[0, 50]$  with initial conditions  $z_1(0) = 1.5 + 0.1d$ ,  $z_2(0) = 0$ , parameter  $\mu = 2 + d$ , and  $N = 200\,000$ .
  - (ii) Computes approximations for  $N \in \{25, 250, 1000, 5000\}$  and plots the first component  $z_1(t)$  of each approximation together with the reference solution in a single figure (overlaid).
  - (iii) Creates a phase portrait ( $z_1$  vs.  $z_2$ ) showing all computed solutions in a single figure (overlaid).

Use axis limits  $t \in [0, 50]$ ,  $z_1 \in [-3, 3]$  for the time-series plot, and  $z_1 \in [-3, 3]$ ,  $z_2 \in [-3\mu, 3\mu]$  for the phase portrait.

**Note:** For very coarse discretizations the explicit Euler method may diverge (producing NaN values). This is expected behavior and not a bug in your code.

# Submission

## Files to Submit

Submit the following Julia source files:

1. `exercise1_LR.jl`
2. `exercise2_QR.jl`
3. `exercise3_interpolation.jl`
4. `exercise4_richardson.jl`
5. `exercise5_newton.jl`
6. `exercise6_euler.jl`

Together with all generated plot images (`.png` files), for example:

- `exercise2_fit.png`
- `exercise3_interpolation.png`, `exercise3_error_n12.png`
- `exercise6_comparison.png`, `exercise6_phase.png`

Upload the folder as a `.zip` to the submission website from **room N14** at <https://na.uni-tuebingen.de/julia-course/submit>. After uploading, **immediately** verify your identity with the course coordinator in the same room by presenting your student ID.