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

due on 28.03.2025

Scientific programming for interdisciplinary mathematics - Worksheet 3

Topics: Numerical integration in 1D, 1D $\mathbb{P}_1$-FEM, sparse matrices

Quadrature — numerical approximation of integrals

For the solution of boundary value problems with the finite element method (FEM), we must compute integrals. Since antiderivatives (and hence exact integrals) can hardly be computed in practice, this is done by *numerical quadrature*: We approximate the integral by an appropriate Riemann sum

$$\int_a^b f(x)\,\mathrm{d}x \approx Q_n(f) := \sum_{j=1}^{n} w_j\, f(x_j),$$

where $a \le x_1 < \cdots < x_n \le b$ are the *quadrature nodes* in the interval $[a,b]$ and $w_1,\dots,w_n \in \mathbb{R}$ are the *quadrature weights*.

Problem 1 (Computing quadrature weights):
We consider the interval $[0,1]$. We fix the quadrature nodes $0 \le x_1 < \cdots < x_n \le 1$. Then, one can compute quadrature weights for the corresponding quadrature $Q_n(f) := \sum_{j=1}^{n} w_j\, f(x_j)$ from the following linear system:

$$Q_n(x^k) = \sum_{j=1}^{n} w_j\, x_j^k \overset{!}{=} \int_0^1 x^k\,\mathrm{d}x = \frac{1}{k+1} \qquad \text{for all } k = 0,\dots,n-1, \tag{1}$$

i.e., we choose the weights in a way that at least all polynomials of degree $\le n-1$ will be integrated exactly (so-called *interpolatory quadrature*). One can show that (1) admits a unique solution $w = (w_1,\dots,w_n) \in \mathbb{R}^n$.

(a) Formulate (1) in terms of a linear system $Aw = b$ that can be solved via the backslash operator in MATLAB and returns the weight vector $w = (w_1,\dots,w_n)$ so that (1) is satisfied.

(b) Write a MATLAB function

`w = quadratureWeights(x)`

which takes the row vector $\mathtt{x} \in \mathbb{R}^n$ of quadrature nodes and returns the row vector $\mathtt{w} \in \mathbb{R}^n$ of the corresponding quadrature weights that satisfy (1).

Problem 2 (Test with equidistant nodes):
For $n \in \mathbb{N}$, use equidistant nodes $x_j := j/n$ for all $j = 0,\dots,n$ and compute the corresponding

weights w_j for all $j = 0, \dots, n$ with your code from Exercise 1. Test the numerical integration of

$$f(x) = \exp(x) \qquad \text{and} \qquad f(x) = x^{3/2}$$

by plotting the error $e_n := |\int_0^1 f(x)\,\mathrm{d}x - Q_n f|$ on the y-axis over the number of quadrature nodes $n = 1, \dots, 20$ on the x-axis in appropriate plots. What rates of convergence do you observe?

Hint: Make the ansatz $e_n = \mathcal{O}(e^{-\alpha n})$ for the first function and $e_n = \mathcal{O}(n^{-\alpha})$ for the second one. In a semilog-plot a logarithmic scaling is used for one axis, while the other axis is linear. In MATLAB, the command `semilogy` plots the logarithm of the y-axis values against the x-axis values. How does a function of the form $e_n = Ce^{-\alpha n}$ look like in a `semilogy` plot?

Problem 3 (Quadrature on arbitrary intervals):
If you have an interpolatory quadrature Q_n on the interval $[0, 1]$, then you can derive an interpolatory quadrature on an arbitrary compact interval $[a, b]$ by means of the substitution formula: Considering the affine transformation $\phi(t) = a + t(b - a)$, we get

$$\int_a^b f(x)\,\mathrm{d}x = (b-a)\int_0^1 f(\phi(t))\,\mathrm{d}t \approx (b-a)\,Q_n(f \circ \phi) = \sum_{j=1}^n (b-a)\,w_j\,f(\phi(x_j)),$$

i.e., defining the quadrature weights $\widetilde{w}_j = (b-a)\,w_j$ and the quadrature nodes $\widetilde{x}_j = \phi(x_j)$, we obtain a quadrature formula $\widetilde{Q}_n$ on $[a, b]$. Write a MATLAB function

$$[\mathtt{x}, \mathtt{w}] = \mathtt{quadrature}(\mathtt{a}, \mathtt{b}, \mathtt{n})$$

which builds on Exercise 1, takes the interval $[a, b]$ and the number of nodes n, and returns row vectors of equidistant quadrature nodes $\mathtt{x} = (x_1, \dots, x_n)$ and quadrature weights $\mathtt{w} = (w_1, \dots, w_n)$ of the corresponding quadrature rule.

Problem 4 (Test of quadrature on arbitrary interval):
Test your implementation from the last exercise to compute the integral

$$\int_{-\pi}^{\exp(1)} \exp(-x)\,\mathrm{d}x \approx 23.07470459693395647.$$

Plot the error on the y-axis over the number of quadrature nodes $n = 1, \dots, 20$ on the x-axis in a semilog-plot.

Problem 5 (Gauss quadrature):
Most important is the so-called *Gauss quadrature* on the compact interval $[-1, 1]$. With n nodes $-1 < x_1 < \ldots < x_n < 1$, it allows to integrate polynomials up to a degree $2n - 1$ exactly, i.e.,

$$Q_n(x^k) = \sum_{j=1}^n w_j\,x_j^k = \int_{-1}^1 x^k\,\mathrm{d}x \quad \text{for all } k = 0, \dots, 2n-1.$$

Moreover, the weights w_j of the Gauss quadrature are positive for all $j = 1, \dots, n$ so that numerical cancellation effects are minimized. Use the affine transformation $\phi(t) = (a + b + t(b - a))/2$ to derive the Gauss quadrature on the compact interval $[a, b]$; cf. Exercise 3. By modifying the code `GaussLegendre.m` found on TUWEL for the Gauss quadrature on $[-1, 1]$, implement a function

$$[\mathtt{x}, \mathtt{w}] = \mathtt{quadratureGauss}(\mathtt{a}, \mathtt{b}, \mathtt{n})$$

that, given the interval $[a, b]$ and the length n, provides the row vectors of the quadrature nodes $\mathtt{x} \in \mathbb{R}^n$ and the quadrature weights $\mathtt{w} \in \mathbb{R}^n$ of the transfered Gauss quadrature on $[a, b]$.

Problem 6 (Test of the Gauss quadrature):
Repeat the test from Exercise 4 with the Gauss quadrature and add the error behaviour into the plot of Exercise 4 for the number of quadrature points $n = 1, \dots, 20$. What do you observe?

Problem 7 (A more involved test case):
Repeat Exercises 4 and 6 for the computation of the integral

$$\int_{-5}^{5} \frac{1}{1 + x^2} \, \mathrm{d}x \approx 2.74680153389003172$$

for the number of quadrature points $n = 1, \dots, 50$. What do you observe in this test case?

Finite element method

We consider the boundary value problem

$$-(a\,u')' + b\,u' + c\,u = f \quad \text{in } [0, 1] \quad \text{subject to} \quad u(0) = 0 = u(1), \tag{2}$$

where $a, b, c \in C[a, b]$ are given coefficient functions and $f \in C[a, b]$ is a given right-hand side. The finite element method (FEM) replaces the Sobolev space $H_0^1(0, 1)$ in the weak formulation by a finite-dimensional subspace $X_h \subset H_0^1(0, 1)$ and seeks $u_h \in X_h$ such that

$$\begin{aligned} \int_0^1 a(x)\, u_h'(x) v_h'(x)\, \mathrm{d}x + \int_0^1 b(x)\, u_h'(x)\, v_h(x)\, \mathrm{d}x + \int_0^1 c(x)\, u_h(x)\, v_h(x)\, \mathrm{d}x \\ = \int_0^1 f(x)\, v_h(x)\, \mathrm{d}x \quad \text{for all } v_h \in X_h. \end{aligned} \tag{3}$$

With a basis $\{\phi_1, \dots, \phi_N\}$ of X_h, we can make the ansatz $u_h = \sum_{j=1}^N x_j\, \phi_j$ with an unknown coefficient vector $x \in \mathbb{R}^N$. Moreover, the discrete variational form (3) is equivalently stated in terms of the linear system

$$(A + B + C)x = L$$

where the matrices $A, B, C \in \mathbb{R}^{N\times N}$ and the vector $L \in \mathbb{R}^N$ read as follows:

$$A_{jk} = \int_0^1 a(x)\,\phi_k'(x)\,\phi_j'(x)\,\mathrm{d}x, \tag{4}$$

$$B_{jk} = \int_0^1 b(x)\,\phi_k'(x)\,\phi_j(x)\,\mathrm{d}x, \tag{5}$$

$$C_{jk} = \int_0^1 c(x)\,\phi_k(x)\,\phi_j(x)\,\mathrm{d}x, \tag{6}$$

$$L_j = \int_0^1 f(x)\,\phi_j(x)\,\mathrm{d}x, \tag{7}$$

for all $j, k = 1, \dots, N$.
In the following, for given nodes $0 = t_1 < t_2 \dots < t_N < t_{N+1} = 1$, we take the hat functions as basis:

$$\phi_j(t) = \begin{cases} \frac{t-t_{j-1}}{t_j-t_{j-1}} & \text{for } t \in [t_{j-1}, t_j], \\ \frac{t_{j+1}-t}{t_{j+1}-t_j} & \text{for } t \in [t_j, t_{j+1}], \\ 0 & \text{for } t \notin [t_{j-1}, t_{j+1}]. \end{cases}$$

Usually, FEMs are implemented by looping over the elements by means of

$$A_{jk} = \int_0^1 a(x)\,\phi_k'(x)\,\phi_j'(x)\,\mathrm{d}x = \sum_{\ell=1}^{N} \int_{t_\ell}^{t_{\ell+1}} a(x)\,\phi_k'(x)\,\phi_j'(x)\,\mathrm{d}x \quad \text{for all } j, k = 1, \dots, N$$

and by noting that a summand is non-zero, only if $\{j, k\} \in \{\ell, \ell+1\}$. Hence, the assembly of A can be done by a loop over ℓ, where, in each step, only 4 entries of the matrix are modified (i.e., updated). Similar observations apply for the right-hand side vector L and the matrices B, C.

Problem 8 (1D FEM for diffusion problem):
We consider the boundary value problem

$$\begin{aligned} -(a\,u')' &= f \quad \text{in } [0,1], \\ u(0) = 0 &= u(1), \end{aligned}$$

where $a \in C[a, b]$ is a given diffusion coefficient and $f \in C[a, b]$.
Completing the template `p1FEM.m` found on TUWEL, write a function

$$\texttt{x} = \texttt{p1FEM}(\texttt{a}, \texttt{f}, \texttt{t})$$

which takes function handles for the functions a and f and a row vector of length $(N+1)$ for the triangulation of $[0, 1]$ and returns a row vector of length $(N+1)$ for the coefficients of the FEM solution u_h with respect to the full basis $\{\phi_1, \dots, \phi_{N+1}\}$ instead of only $\mathcal{B}_0^1 := \{\phi_2, \dots, \phi_N\}$ (i.e., $x_1 = 0 = x_{N+1}$). **All integrals must be computed by numerical quadrature** (use, e.g., your code from Exercise 5). Note that we assemble the matrix and the right-hand side for all hat-functions $\phi_1, \dots, \phi_{N+1}$, while the FEM system corresponds only to the $(N-1) \times (N-1)$ subsystem associated with $\phi_2, \dots, \phi_N$. In MATLAB, solving the FEM system is done via

```
x(2:N) = A(2:N, 2:N)\L(2:N);
```

Test your implementation from Exercise 8 for $a(x) = 1$ and $f(x) = 1$ for all $x \in [0, 1]$ by comparing it to the exact solution $g(x) = \frac{1}{2}(1-x)x$ in the same plot. Use an equidistant triangulation, e.g., `t = linspace(0, 1, 100)`. Note that this should reproduce your results from NumPDE Exercise 2.3!

Problem 9 (1D FEM for general second order elliptic ODEs):
Extend your code `p1FEM.m` to cover general second order linear elliptic boundary value problems (2), i.e., extend your code by also computing the B and the C matrix from (5)–(6). The function should be called by

$$\mathtt{x} = \mathtt{p1FEM}(\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{f}, \mathtt{t})$$

taking function handles for the functions a, b, c, and f and a row vector of length $(N+1)$ for the triangulation $\mathcal{T}$ of $[0, 1]$.

Problem 10 (Diffusion-advection-reaction test case):
Test your implementation from Exercise 9 for $a(x) = 1$, $b(x) = \text{const}$, $c(x) = -1$ and $f(x) = 1$. Choose the constant function $b(x) \in \{-10, -5, 0, 5, 10\}$. What role does b play for the solution?

Problem 11 (Tracking the computational time, storage reduction with sparse matrices):
For various $N = 2^n$ and uniform triangulations $\mathcal{T}_N$ with nodes $t_j := j/N$ for all $j = 0, \dots, N$, measure the computational time of the call of `p1FEM` from Exercise 8 (or Exercise 9 or your implementation from NumPDE Exercise 2.3) by use of `tic` and `toc`. Plot the computational time T_N on the y-axis over N on the x-axis in a loglog-plot. What growth $T_N = \mathcal{O}(N^\alpha)$ do you observe?
Note that your FEM matrix $S := A + B + C$ is tridiagonal and hence sparse (i.e., only $\mathcal{O}(N)$ of the overall $\mathcal{O}(N^2)$ entries are non-zero). Make yourself familiar with the MATLAB command `sparse`. How could this be used to improve the function `p1FEM`? You don't need to implement it, but you should be able to explain it.

Take-home messages

(i) A centerpiece of this course is the numerical solution of partial differential equations which requires to compute integrals by means of quadrature (i.e., numerical integration). However, the stability of the quadrature depends heavily on the selection of the evaluation points. While equidistant quadrature nodes seem natural, the Gauss quadrature is more stable and accurate.
(ii) An algebraic error decay $e_n = \mathcal{O}(n^{-\alpha})$ is best visualized in a double logarithmic plot with e_n on the y-axis over n on the x-axis (`loglog` in MATLAB)), where the graph is a straight line with slope $-\alpha$. An exponential decay $e_n = \mathcal{O}(e^{-\alpha n})$ is best visualized with $\log(e_n)$ on the y-axis over n on the x-axis (`semilogy` in MATLAB), where the graph is a straight line with slope $-\alpha$.
(iii) Finally, the exercise sheet culminates in the implementation of the $\mathbb{P}_1$-FEM for the numerical solution of a boundary value problem in 1D.