1. **[Extrapolation, 3+2pt]** Extrapolation is interpolation where the point of interest, at which we evaluate the interpolation polynomial, is outside the interpolation interval.\(^1\) Extrapolation can, for instance, be used to approximate limits. Let us compute the sum

\[
s := \sum_{k=1}^{\infty} \frac{1}{k^2},
\]

whose exact value is \(\pi^2/6\), through extrapolation. For that purpose, consider the partial sums

\[
s_i := \sum_{k=1}^{n_i} \frac{1}{k^2} \quad \text{with} \quad n_i = 10 \cdot 2^i, \quad \text{for} \quad i = 1, \ldots, 5.
\]

Denoting by \(h_i := 1/n_i\), extrapolation then amounts to fitting a polynomial to the values \((h_i, s_i)\), for \(i = 1, \ldots, 5\), and evaluating that polynomial at \(h = 0\) (which corresponds to \(n = \infty\)). The result \(\tilde{s}\) is the extrapolated value for the approximation of \(s\).

(a) Use build-in functions for the interpolation/extrapolation (i.e., in MATLAB, the function `polyfit` and `polyval`). List the values \(s_i\) and the extrapolated value \(\tilde{s}\) and highlight the number of exact digits. Note that you will have to output at least 12 digits (use `format long` in MATLAB).

(b) Extrapolation is one of the problems where an interpolating polynomial only has to be evaluated at a single point, something the Aitkin algorithms can be used for. Use the Aitkin algorithm instead of the build-in polynomial interpolation functionality.

2. **[Hermite quadrature, 3+3pt]** Here, we study a quadrature rule that involves derivatives.

(a) Give the Lagrange basis polynomials \(H_0(t), H_1(t), H_2(t) \in P_2\) for Hermite interpolation of \(f(0), f'(0)\) and \(f(1)\), i.e., find the quadratic polynomials that satisfy

\[
\begin{align*}
H_0(0) &= 1, & H_0'(0) &= 0, & H_0(1) &= 0, \\
H_1(0) &= 0, & H_1'(0) &= 1, & H_1(1) &= 0, \\
H_2(0) &= 0, & H_2'(0) &= 0, & H_2(1) &= 1.
\end{align*}
\]

A simple way to derive these is using the Newton basis and divided differences.

\(^1\)Compare with Section 9.4.2 in Deuflhard/Hohmann or the Wikipedia page: [https://en.wikipedia.org/wiki/Extrapolation](https://en.wikipedia.org/wiki/Extrapolation).
(b) For numerical integration on the interval \([0, 1]\), find weights \(\alpha_1, \alpha_2, \alpha_3\) such that the quadrature formula
\[
\hat{I}(f) := \alpha_1 f(0) + \alpha_2 f'(0) + \alpha_3 f(1)
\]
integrates polynomials \(p \in P_2\) exactly. Hint: Use that the Hermite polynomials \(H_0, H_1, H_2\) are a basis of \(P_2\), i.e., it suffices if \(\hat{I}\) is exact for these \(H_i\).

3. [Trapezoidal and Simpson sum, 4+2+2pt] Write functions \texttt{trapez}(f,x) and \texttt{simpson}(f,x) to approximate integrals
\[
I(f) = \int_a^b f(t) \, dt
\]
using the trapezoidal and Simpson’s rules on each sub-interval \(I_i := [x_i, x_{i+1}]\), \(i = 0, 1, \ldots, N - 1\), where we assume that these sub-intervals cover \([a,b]\) and have no overlap except the sub-interval boundary points \(x_i\). The input vector to the functions should be \(x = (x_0, \ldots, x_N)\), i.e., the vector of sub-interval boundaries, and \(f\) is either a function handle\(^2\) or the vector \(f = (f(x_0), \ldots, f(x_N))\). Note that \texttt{simpson()} also requires the function values at the mid points of each interval, i.e., if \(f\) is a vector, it must include the values \(f((x_{i+1} + x_i)/2)\) and must thus be of size \(2N + 1\).

(a) Hand in code listings of these functions, and use them to approximate the integrals
\[
\int_0^1 x^4 \, dx, \quad \int_0^1 \sqrt{x} \, dx.
\]
Compare the numerical errors for uniformly spaced points \(x_i = ih, \ i = 0, \ldots, N, \ h = 1/N\) for both quadrature rules. Try different \(N\) and plot the quadrature errors versus \(N\) in a double-logarithmic plot.

(b) Estimate the error behavior by fitting \(cN^\kappa\), with \(c, \kappa \in \mathbb{R}\) to the quadrature errors. To avoid having to solve a nonlinear least squares problem for \(c\) and \(\kappa\), apply the logarithm to \(cN^\kappa\) and solve a \textit{linear} least squares problem for the parameters \(d := \log(c)\) and \(\kappa\).

(c) Propose non-uniform interval points \(x_i\) to faster approximate \(\int_0^1 \sqrt{x} \, dx\), and plot your numerical result for different \(N\). Again, estimate \(d\) and \(\kappa\).

4. [Hierarchical Lobatto quadrature, 3+2pt] Gauss\(^3\) quadrature chooses the quadrature point location and the quadrature weights such that a high order approximation is achieved with a minimal number of points. The Gauss quadrature points for an interval \([a, b]\) do not include the beginning and end points \(a\) and \(b\). It is often convenient if these points are included. When not all points are allowed to vary but the location of some points is fixed, the corresponding quadrature rules are called Gauss-Lobatto rules. Let us compute such


\(^3\)In this example, we use “Gauss quadrature” short for Gauss-Legendre quadrature, since we use the integral weight function \(\omega \equiv 1\), and the corresponding orthogonal polynomials are the Legendre polynomials.
Gauss-Lobatto points and corresponding quadrature weights for the interval \([-1, 1]\). Due to the symmetry, the quadrature formula for 4 nodes is:

\[
\hat{I}_1(f) = \omega_1(f(-1) + f(1)) + \omega_2(f(-t_2) + f(t_2)),
\]

with to be determined weights \(\omega_1, \omega_2\) and the interior node location \(t_0\). Because of symmetry, the monomials with odd degree are integrated exactly by the above formula. Since there are 3 parameters, we hope that we can choose them to integrate the even monomials \(1, x^2, x^4\) exactly.

(a) Give the set of nonlinear equations needed to be satisfied by \(\omega_1, \omega_2\) and \(t_0\) to integrate the first three even monomials exactly, and solve it. You should be able to do this analytically.\(^4\)

(b) Often, one is interested in an adaptive quadrature rule that allows to add quadrature points while re-using the quadrature points from a less accurate rule, where the function has already been evaluated.\(^5\) Thus, let us try to extend the above quadrature rule by adding internal nodes, while re-using the point \(t_2\) from above. We will add three points: Due to symmetry, one must lie at the center \((t = 0)\), and the other two added points must be symmetric around the center, such that the new quadrature rule becomes:

\[
\hat{I}_2(f) = \omega_3(f(-1) + f(1)) + \omega_4(f(-t_2) + f(t_2)) + \omega_5(f(-t_3) + f(t_3)) + \omega_6 f(0).
\]

Note that \(t_2\) is fixed but the corresponding weight \(\omega_4\) can be different compared to the quadrature formula \(\hat{I}_1\). Thus, the free variables in \(\hat{I}_2\) are \(\omega_3, \omega_4, \omega_5, \omega_6\) and \(t_3\), and we can hope to integrate the monomials \(1, x^2, x^4, x^6, x^8\) exactly. Specify the corresponding nonlinear system.

(c) [2pt extra credit] Solve this nonlinear system for the five parameters using Newton’s method.

\(^4\)Note that a Gauss rule with 4 points and 4 weights allows to integrate polynomials up to order 7 exactly. This Gauss-Lobatto (or Gauss-Legendre-Lobatto) rule, which fixes two node locations, only integrates polynomials up to order 5 exactly. This, namely that Gauss quadrature is two orders more accurate than Lobatto-Gauss quadrature, remains true also for higher-order quadrature.

\(^5\)Function evaluation is usually the most expensive step in numerical quadrature, i.e., the target is to minimize the number of required evaluations of \(f\) while obtaining a good approximation to the integral.