AM205 HW3. Numerical calculus

P1. Adaptive integration [23 pts]

(a) [4 pts] Given the cubic Legendre polynomial $P_3(x) = \frac{1}{2}x(5x^2 - 3)$, derive the 3-point Gauss quadrature rule on the interval $[-1, 1]$ by evaluating the relevant integrals by hand or symbolically. Demonstrate that this quadrature rule integrates all polynomials up to the expected degree exactly.

(b) [12 pts] In the lectures we discussed a method of evaluating integrals $\int_a^b f(x)dx$ by adaptively refining the calculation in regions where the function varies rapidly. To begin, a tolerance level $T > 0$ is introduced, and the calculation starts using a single integration interval $[a, b]$.

Let $I_{a,b}$ be the integral of f over this interval using the three-point quadrature rule from part **(a)**, which needs to be rescaled to apply to $[a, b]$ instead of $[-1, 1]$. In addition, define $c = \frac{a+b}{2}$ and calculate $\hat{I}_{a,b} = I_{a,c} + I_{c,b}$ as a more refined estimate of the integral. Then $E_{a,b} = |I_{a,b} - \hat{I}_{a,b}|$ is an estimate of the error of $I_{a,b}$. If $E_{a,b} < (b - a)T$, then the error is acceptable, and the method can terminate. Otherwise this interval must be subdivided into $[a, c]$ and $[c, b]$, and the above procedure must be applied to these two intervals. The procedure must be applied recursively until the errors become smaller than the tolerance.

Write a function to implement this adaptive integration scheme. Using $T = 10^{-6}$, apply it to the integrals

$$\int_{-1}^{7/2} (-x^m - (x + 3)^2 + 4)dx$$

for $m = 4,\ 5,\ 6,\ 7,\ 8$. For each case, report the value of the integral, the total estimated error (by summing the relevant $E_{\alpha,\beta}$ terms), and the total number of intervals that are used.

(c) [7 pts] Use your adaptive integration routine from part **(a)** and $T = 10^{-6}$ to evaluate the three integrals

$$\int_{-1}^{3} |x - \frac{\pi}{8}|dx, \qquad \int_{-1}^{1} g(x)dx, \qquad \int_{-1}^{1} x \sin \frac{x}{x^2 + 0.01}\, dx.$$

where

$$g(x) = \begin{cases} 4x^5 + 12x^4 - 8x^2 - x, & \text{if } x < 0, \\ \sin(5x^2), & \text{if } x \geq 0. \end{cases}$$

For each case, report the value of the integral, the total estimated error (by summing the relevant $E_{\alpha,\beta}$ terms), and the total number of intervals that are used. For each integral, plot the integrand and mark the quadrature points and subinterval endpoints.

P2. Finite differences and least squares [23 pts]

Here you will consider two approaches to derive a finite difference approximation: from the Taylor expansion and by differentiating a polynomial fitted to the function values.

(a) [2 pts] Recall that we can numerically approximate $f'(x)$ via the Taylor expansion of a function f about x. In the simplest case, we evaluate the expansion at $x + h$ (or $x - h$) and neglect an $\mathcal{O}(h^2)$ term to obtain either the "forward difference formula" or the "backward difference formula". We can combine these approximations to obtain the "centered difference formula" which is second-order accurate.

Using the same approach, derive a fourth-order approximation to $f'(x)$. Evaluate the Taylor expansion of f about x at $x + h$, $x - h$, $x + 2h$, $x - 2h$ and then combine these formulas to cancel out the terms of order $\mathcal{O}(h^2)$, $\mathcal{O}(h^3)$, and $\mathcal{O}(h^4)$.

(b) [4 pts] For each of the following functions

- $f(x) = e^x$, for $x \in [0, 2]$
- $f(x) = (\sin x)^2$, for $x \in [0, \pi]$
- $f(x) = x^5 - 3x^3$, for $x \in [0, 2]$

evaluate the second-order centered difference and fourth-order approximation derived in part **(a)** and plot their absolute error in the log-linear scale. Perform your analysis for $h = 0.2$ and $h = 0.02$. Qualitatively describe your results and explain your observations. Does the error reduce according to the expected order of accuracy?

(c) [8 pts] In the remainder of this problem, you will construct a family of finite difference approximations using least-squares fitting. Consider a general finite difference approximation to the first derivative

$$f'(x) \approx \frac{w_0 f(x - h\frac{m-1}{2}) + w_1 f(x - h\frac{m-1}{2} + h) + \ldots + w_{m-1} f(x + h\frac{m-1}{2})}{h},$$

which is a linear combination of function values on a stencil of m equidistant points centered at x. The coefficients $w \in \mathbb{R}^m$ determine the approximation. Design an algorithm to numerically compute these coefficients by differentiating a polynomial of degree $p \leq m + 1$ fitted to the function values. Note that the coefficients w will only depend on m and p. They should not depend on h, f, or x. Your algorithm should construct the Vandermonde matrix for the stencil points, solve the normal equations to find the least-squares fit, and then exactly evaluate its derivative. Implement the algorithm.

(d) [3 pts] Numerically compute the coefficients w and list them (up to four significant digits) for the following cases

- $(m, p) = (3, 2), (5, 4)$
  This polynomial interpolates all of m function values ($m = p + 1$). The approximations should coincide with centered differences in part **(a)**.
- $(m, p) = (5, 2), (7, 4)$
  These have two extra points ($m = p + 3$) and will have some smoothing effect.
- $(m, p) = (7, 2), (9, 4)$
  These have four extra points ($m = p + 5$) and will have a stronger smoothing effect.

(e) [6 pts] Consider a function

$$f(x) = \sin(10x^2)$$

and the same function perturbed with "pseudo-random" noise

$$g(x) = f(x) + 0.001\cos(1000\cos(1000(x+1))$$

in the range $x \in [0, 1]$. First, take $p = 2$ and use your algorithm from part **(c)** to obtain three approximations $\hat{g}'_m(x)$ to $g'(x)$ for $m = 3$, 5, and 7 points with the step size $h = 0.01$. Evaluate the approximations at 101 equidistant points in $[0, 1]$. Plot the derivative of $f'(x)$ (not the perturbed $g'(x)$) and your three approximations obtained from $g(x)$. Plot the absolute error $|\hat{g}'_m(x) - f'(x)|$ in the log-linear scale. Report the root-mean-square (RMS) error in $|\hat{g}'_m(x) - f'(x)|$. Repeat the same for $p = 4$ and $m = 5,\ 7,\ 9$. Which of the approximations gives the lowest RMS error? Do you observe different trends for the RMS error in the cases $p = 2$ and $p = 4$ as m increases? Explain why.

P3. Gravity assist [23 pts]

Here you will navigate a spacecraft between two planets to reach a target. Treat the spacecraft as a particle $\mathbf{x}(t) \in \mathbb{R}^2$ and the two planets as stationary point masses at $\mathbf{p}_1 = (-0.5, 0)$ and $\mathbf{p}_2 = (0.5, 0)$. The spacecraft moves according to Newton's second law

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = \mathbf{f}(\mathbf{x})$$

under the gravitational acceleration toward both planets

$$\mathbf{f}(\mathbf{x}) = \frac{\mathbf{p}_1 - \mathbf{x}}{|\mathbf{p}_1 - \mathbf{x}|^3} + \frac{\mathbf{p}_2 - \mathbf{x}}{|\mathbf{p}_2 - \mathbf{x}|^3}\,.$$

The spacecraft starts at point $\mathbf{x}_0 = (0, -1)$ with an initial velocity $\mathbf{v}_0 = (u_0, v_0)$. The goal will be to find an initial velocity such that the spacecraft reaches a target $\mathbf{x}_T = (0, 1)$ at time $T = 1$.

(a) [6 pts] Solve the initial value problem for the motion of the spacecraft starting at $\mathbf{x}(0) = \mathbf{x}_0$ in the range $t \in [0, T]$ for the following cases

(i) Vary u_0 over 11 equidistant points in $[-0.1, 0.1]$ and set $v_0 = 1.15$.
These trajectories should approach the target passing $\mathbf{p}_2$ on the left.

(ii) Vary u_0 over 11 equidistant points in $[1.5, 1.7]$ and set $v_0 = 1.15$.
These trajectories should approach the target passing $\mathbf{p}_2$ on the right.

You can use a library function to solve the problem (e.g. `scipy.integrate.solve_ivp`). Plot the trajectories and mark the starting point, target, and planets.

(b) [12 pts] In this part, you will find the initial velocity by solving a nonlinear boundary value problem. The boundary conditions are $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{x}(T) = \mathbf{x}_T$. The initial velocity is unknown. Represent the approximate trajectory as $\mathbf{x}_i \approx \mathbf{x}(t_i)$ on a grid of $N+1$ equidistant points $t_i \in [0, T]$ (including the endpoints) for $i = 0, \ldots, N$ with $N = 100$. Use the following discretization

$$\frac{\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{\Delta t^2} = \mathbf{f}(\mathbf{x}_i), \quad i = 1, \ldots, N-1,$$

where $\Delta t = T/N$. Note that $\mathbf{f}$ is a nonlinear function, so the discretized equation is nonlinear with respect to $\mathbf{x}_i$. We will solve the problem iteratively using Newton's method. Let $\mathbf{x}_i^s$ be the trajectory at iteration s and linearize the equation about $\mathbf{x}_i^s$

$$\frac{\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1}}{\Delta t^2} = \mathbf{f}(\mathbf{x}_i^s) + \frac{\partial \mathbf{f}(\mathbf{x}_i^s)}{\partial \mathbf{x}}(\mathbf{x}_i - \mathbf{x}_i^s), \quad i = 1, \ldots, N-1.$$

Solve this linear problem for $\mathbf{x}_i$ using any linear solver of your choice. One option is to adapt the scalar TDMA algorithm (e.g. `[spline_tdma.py]`): scalar unknowns will turn into vectors $\mathbf{x}_i$, coefficients will turn into 2×2 matrices, and division will turn into matrix inversion. After solving the linear system, update the trajectory as $\mathbf{x}_i^{s+1} = \mathbf{x}_i$ and proceed to the next iteration. Iterate until the maximum absolute change is smaller than a given tolerance, i.e. $\max_{i=0,\ldots,N} |\mathbf{x}_i^{s+1} - \mathbf{x}_i^s| < 10^{-10}$. Implement the method.

(c) [5 pts] The final trajectory in part **(b)** will depend on the initial guess $\mathbf{x}_i^0$ at iteration $s = 0$. Consider two cases corresponding to (i) and (ii) from part **(a)** and pick one trajectory obtained there as an initial guess. Plot the initial guess, trajectories after the first three iterations, the final trajectory, and also mark the starting point, target, and planets. Report the approximate initial velocity $(\mathbf{x}_1 - \mathbf{x}_0)/\Delta t$ from the final trajectory. Report the number of iterations to achieve convergence.

P4. Advection and anti-diffusion [23 pts]

Here you will amend a discretization of the linear advection equation with an anti-diffusion term to improve its accuracy. The problem consists of the linear advection equation

$$u_t + u_x = 0$$

for the unknown function $u(x,t)$ in the domain $x \in [0,1]$ and $t \in [0,1]$ with periodic boundary conditions. Consider two choices of the initial conditions

$$u_{\text{smooth}}(x) = \begin{cases} \exp\left(1 - \frac{1}{1-z^2}\right), & |z| < 1 \\ 0, & \text{otherwise} \end{cases}$$

$$u_{\text{step}}(x) = \begin{cases} 1, & |z| < 1 \\ 0, & \text{otherwise} \end{cases}$$

where $z(x) = (x - 0.5)/0.2$. Use the method of lines to solve the problem, i.e. reduce the equation to a system of ODEs, and use a library function to solve the ODEs (e.g. `scipy.integrate.solve_ivp`). Represent the approximate solution $u_i(t) \approx u(x_i, t)$ on a grid of N equidistant points $x_i = ih$ for $i = 0, \ldots, N-1$ with $h = 1/N$ and $N = 100$. To impose periodic boundary conditions, define $u_{-1} = u_{N-1}$ and $u_N = u_0$.

(a) [7 pts] Implement the first-order upwind scheme

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} + \frac{u_i - u_{i-1}}{h} = 0.$$

Consider two choices of the initial conditions $u(x,0) = u_{\text{smooth}}(x)$ and $u(x,0) = u_{\text{step}}(x)$. For each case, solve the problem until $t = 1$ and plot $u_i(0)$ together with $u_i(1)$. In the exact solution, the advected profile coincides with the initial profile since it passes exactly one period. Qualitatively describe the approximate solutions in both cases. Do you observe smearing, oscillation, change of magnitude, or phase shift?

(b) [7 pts] Use the Taylor expansion to find $A > 0$ in the leading term of the truncation error

$$\frac{u_i - u_{i-1}}{h} = u_x - Ahu_{xx} + \mathcal{O}(h^2).$$

Note that the approximation from part **(a)** can be viewed as a solution of the following equation

$$u_t + u_x = Ahu_{xx} + \mathcal{O}(h^2)$$

called the *modified equation*, which looks like the original advection equation with an added diffusion term. One natural idea to improve the accuracy of the discretization is to subtract an approximation of the leading error term. Consider another scheme

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} + \frac{u_i - u_{i-1}}{h} = -A\frac{u_{i+1} - 2u_i + u_{i-1}}{h}$$

which is now second-order accurate. The added term is an *anti-diffusion* term and it is supposed to counteract numerical diffusion. Implement the new scheme and plot the solutions at $t = 1$ for both choices of the initial conditions. Do you observe smearing, oscillation, change of magnitude, or phase shift?

(c) [9 pts] Modify the previous scheme by only applying the anti-diffusion term if the solution is monotonic

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} + \frac{u_i - u_{i-1}}{h} = -G(u_{i-1}, u_i, u_{i+1}) A \frac{u_{i+1} - 2u_i + u_{i-1}}{h},$$

where function

$$G(v_0, v_1, v_2) = \begin{cases} 1, & (v_2 - v_1)(v_1 - v_0) > 0 \\ 0, & \text{otherwise} \end{cases}$$

detects if the sequence v_0, v_1, v_2 is monotone. Implement the scheme, compute the solutions for the same two cases and plot the results. The new scheme should reduce the effect of numerical diffusion without producing oscillatory solutions.