AM205 HW4. PDEs, nonlinear equations, optimization

P1. Finite differences for advection [31 pts]

Here you will compare the stability and accuracy of two finite difference schemes for the linear advection equation $u_t + cu_x = 0$. Assume that $c \in \mathbb{R}$ is a constant and $c > 0$. The exact solution $u(x,t)$ is approximated with values $u_j^n \approx u(x_j, t^n)$ on a uniform grid $x_j = j\Delta x$ and $t^n = n\Delta t$. Denote the CFL number as $\nu = c\Delta t/\Delta x$. The two schemes are the central difference scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \tag{1}$$

and the second-order upwind scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{3u_j^n - 4u_{j-1}^n + u_{j-2}^n}{2\Delta x} = 0. \tag{2}$$

(a) [2 pts] For each scheme, determine the range of ν that satisfy the CFL condition.

(b) [8 pts] For each scheme, show that its truncation error has the form

$$T_j^n = A_1 u_{tt}(t^n, x_j)\Delta t + A_2 u_{xxx}(t^n, x_j)\Delta x^2 + \text{h.o.t.} \tag{3}$$

and provide expressions for the factors A_1 and A_2.

(c) [11 pts] To analyze the stability, substitute the ansatz $u_j^n = \lambda^n e^{ijk\Delta x}$ into each scheme and derive an expression for the amplification factor λ as a function of $k\Delta x$ and ν. Stability implies that the Fourier modes are not amplified, i.e. $|\lambda| \le 1$. However, you are not asked to solve this inequality analytically. Instead, perform the following for each scheme

- Set $\nu = 0.25$ and plot $|\lambda|$ as a function of $k\Delta x \in [0, \pi]$. Report $k\Delta x$ and $|\lambda|$ of the "most unstable" mode, i.e. corresponding to the largest $|\lambda|$. Are there any values of $k\Delta x$ for which the scheme is stable? Use samples of λ obtained at 100 equidistant points $k\Delta x \in [0, \pi]$.
- Plot $\max_{k\Delta x} |\lambda|$ as a function of $\nu \in [0, 0.5]$. Are there any values of ν for which the scheme is stable? Use samples of λ obtained at 100 equidistant points $k\Delta x \in [0, \pi]$ and 100 equidistant points $\nu \in [0, 0.5]$.

(d) [10 pts] Implement both schemes and solve an initial value problem for the advection equation with periodic boundary conditions in the range $x \in [0,1]$ using a uniform grid of 128 points. Set $c = 1$ and $\nu = 0.25$. You can start from `[examples/unit3/advection.py]` which already implements the central difference scheme. Try using a smooth initial profile $u(x,0) = e^{-20(x-0.5)^2}$ and solve the equation in the time range $t \in [0, 0.5]$. Also try using a discontinuous initial profile $u(x,0) = 1$ if $|x - 0.5| < 0.2$ and $u(x,0) = 0$ otherwise and solve the equation in the time range $t \in [0, 0.1]$. These parameters are chosen to demonstrate that although both schemes can be unstable and have the same order of accuracy, the second-order upwind scheme produces more accurate results.

P2. Newton fractal [31 pts]

A fractal is a geometric shape containing structures at arbitrarily small scales. One example is the Newton fractal which is characterized by Newton's method applied to a complex-valued polynomial. Following the canonical example of the Newton fractal, we use the polynomial

$$f(z) = z^3 - 1 \tag{4}$$

with complex roots

$$1, \quad -\tfrac{1}{2} + i\tfrac{\sqrt{3}}{2}, \quad -\tfrac{1}{2} - i\tfrac{\sqrt{3}}{2}. \tag{5}$$

A unique color is assigned to each root (e.g. red, green, and blue). The fractal is represented by an image in the complex plane. Each point $z \in \mathbb{C}$ is colored depending on the outcome of Newton's method applied to f and started from z. If the iteration converges to a root, z is colored by the color of that root. Otherwise, z receives a default color (e.g. white). You will visualize the canonical Newton fractal and also apply the same technique to other iterative methods.

Denote $z = x + iy \in \mathbb{C}$, $f(z) = u(x,y) + iv(x,y) \in \mathbb{C}$, and $\mathbf{x} = (x,y) \in \mathbb{R}^2$. Define a vector-valued function $\mathbf{g} : \mathbb{R}^2 \to \mathbb{R}^2$ as $\mathbf{g}(\mathbf{x}) = (u(x,y), v(x,y))$. These types of notation will be used interchangeably. Recall that the function $f(z)$ is complex differentiable if partial derivatives of $u(x,y)$ and $v(x,y)$ exist and satisfy the Cauchy–Riemann equations $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$ and $\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$.

(a) [6 pts] Newton's method can be formulated in two different ways: for the complex-valued function $f(z)$ and the vector-valued function $\mathbf{g}(\mathbf{x})$. They correspond to two update rules

$$z_{k+1} = z_k - f(z_k)/f'(z_k) \tag{6}$$

and

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J_g}(\mathbf{x_k}))^{-1}\mathbf{g}(\mathbf{x}_k), \tag{7}$$

where $\mathbf{J_g}$ is the Jacobian matrix of $\mathbf{g}$. Analytically show that both formulations are equivalent, i.e. they produce the sequences related as $z_k = x_k + iy_k$. To do this, use the Cauchy–Riemann equations and write both update rules as linear equations instead of inverting the derivatives. Your proof should be general for any complex differentiable function $f(z)$, e.g. without assuming that it is a polynomial.

(b) [8 pts] Implement Newton's method for $f(z)$ given by (4). Terminate the algorithm if any of the following conditions is met

- the distance to one of the roots z is below the tolerance $\epsilon = 10^{-3}$, i.e. $|z - z_k| < \epsilon$;
- the absolute value exceeds the threshold $A = 10^3$, i.e. $|z_k| > A$;
- the number of iterations reaches $K = 100$, i.e. $k \geq K$.

Plot the trajectories of the method, i.e. all points z_k, for each of the starting points

$$-0.2 + 0.75i, \quad 0.4 + 0.75i, \quad 1 + 0.75i, \quad 1.6 + 0.75i, \tag{8}$$

which are chosen to demonstrate convergence to different roots. Mark the starting points and the exact roots (5) on the plot.

Generate an image representing the Newton fractal. The image should cover the rectangle $x, y \in [-2, 2]$ and have a resolution of 256×256. You may increase or decrease the resolution depending on your implementation's performance. Mark the exact roots (5) on the image. Qualitatively describe your observations. Does the method converge from any starting point?

(c) [8 pts] Recall the secant method which replaces the derivative in Newton's method by its finite-difference approximation using the function values from two previous iterations. The secant method is only applicable to univariate scalar functions, since in a higher-dimensional space the derivative is a matrix. However, the secant method can be applied on the complex plane using the following update rule

$$z_{k+1} = z_k - f(z_k)\frac{z_k - z_{k-1}}{f(z_k) - f(z_{k-1})}. \tag{9}$$

Repeat part **(b)** using the secant method. For the first iteration to obtain z_1 given z_0, apply Newton's method.

(d) [9 pts] Another method of root finding is gradient descent applied to the squared residual

$$L(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|_2^2 \tag{10}$$

with the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla L(\mathbf{x}_k), \tag{11}$$

where $\eta = 0.025$. Repeat part **(b)** using this method. The gradient $\nabla L(\mathbf{x})$ should be computed analytically (symbolically or by hand).

P3. Himmelblau's function [30 pts]

One benchmark for optimization algorithms is the minimization of Himmelblau's function

$$f(x,y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2, \tag{12}$$

which has four global minima of 0. You will apply three different optimization algorithms to this function with various starting points aiming for its four different global minima. Denote $\mathbf{x} = (x, y)$. You should not use library functions that already implement the three considered optimization algorithms. However, you may and should use library functions for specific steps of the algorithms, as indicated below.

(a) [10 pts] Minimize Himmelblau's function using gradient descent with a constant factor, i.e. with the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k), \tag{13}$$

where $\eta = 0.015$. Try two sets of starting points

- $(1, 1)$, $(-1, 1)$, $(-1, -1)$, $(1, -1)$,
- $(3, 3)$, $(-3, 3)$, $(-3, -3)$, $(3, -3)$.

Terminate the algorithm if either $\|\nabla f(\mathbf{x}_k)\|_2 < \epsilon$ with $\epsilon = 10^{-5}$ or the number of iterations reaches 1000. For each set of the starting points, plot in the same graph

- contours of $f(x,y)$ with $x, y \in [-5, 5]$,
- starting points,
- trajectories of the optimizer, i.e. points $\mathbf{x}_k$ from all iterations.

For each starting point, report the performed number of iterations, solution $\mathbf{x}_{\text{term}}$ at the last iteration, values of the function $f(\mathbf{x}_{\text{term}})$, and its gradient $\nabla f(\mathbf{x}_{\text{term}})$. Qualitatively describe your observations. Does the method converge to the global minimum? Does the method find critical points, i.e. such that $\nabla f(\mathbf{x}) = 0$?

(b) [10 pts] Repeat part **(a)** using steepest descent method, i.e. with the update rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{s}, \tag{14}$$

where $\mathbf{s} = -\nabla f(\mathbf{x}_k)$ is the line search direction, and α minimizes the function

$$g(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{s}). \tag{15}$$

Use any optimizer to minimize the function $g(\alpha)$. One option is the BFGS method available in `scipy.optimize.minimize`. Another option is `scipy.optimize.line_search` that enforces strong Wolfe conditions.

(c) [10 pts] Repeat part **(a)** using Newton's method, i.e. with the update rule

$$H_f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\nabla f(\mathbf{x}_k), \tag{16}$$

where $H_f(\mathbf{x}_k)$ is the Hessian of f. Use a library function to solve the linear system.