Homework 4

Exercise 1: solving the 2D heat equation

Consider the following PDE

$$
\begin{aligned}
u_t - \Delta u &= 2\pi\left[4\pi\cos(2\pi t) - \sin(2\pi t)\right]\sin(2\pi x)\sin(2\pi y), && \forall \mathbf{x} \in \Omega = (0,1)^2, && \text{(1a)}\\
u(\mathbf{x},0) &= u_0(\mathbf{x}) = \sin(2\pi x)\sin(2\pi y), && \forall \mathbf{x} \in \Omega = (0,1)^2, && \text{(1b)}\\
u(\mathbf{x},t) &= 0 && \forall \mathbf{x} \in \partial\Omega. && \text{(1c)}
\end{aligned}
$$

Our objective is to obtain the solution at $t = 1$. The exact solution for this problem is given by

$$u = \cos(2\pi t)\sin(2\pi x)\sin(2\pi y)$$

Proceed as follows:

* Considering backward Euler, obtain the discrete weak formulation of (1).
* Project the initial condition to obtain $u_h(\mathbf{x},0)$. Alternatively, you can find $u_h(\mathbf{x},0)$ as follows

  $$u_h(\mathbf{x},0) = \sum_j U_j^0 \varphi(\mathbf{x}),$$

  where $U_j^0 = u_0(\mathbf{x}_j)$.
* Use the Matlab code to obtain the soltion at $t = 1$ with either quadrilateral or triangular elements. Consider continuous piecewise linear polynomials. Plot and report the solution at $t = 0.25, 0.5, 0.75$ and 1.

Hint: for each time step, we know U^n and we want to find

$$u_h(\mathbf{x},t^{n+1}) = \sum_j U_j^{n+1}\varphi(\mathbf{x}).$$

Therefore, we must obtain U^{n+1}. Express the problem as a linear algebra problem $AU^{n+1} = F$, where A is a combination of the mass and the stiffness matrices. Solve for U^{n+1} and proceed to the next time step.

About the boundary conditions: the Dirichlet boundary conditions can be imposed by modifying the matrix A as follows. Let i be a boundary node, then impose zeros in all entries $j \neq i$ and one in $j = i$. For the right hand side F, impose $F_i = u(\mathbf{x}_i, t) = 0$.

Exercise 2: solving the 2D advection equation via the Galerkin method

Consider the 2D advection equation

$$u_t + \mathbf{a} \cdot \nabla u = 0, \qquad \forall \mathbf{x} \in (0,1)^2, \tag{2a}$$

$$u(\mathbf{x}, 0) = \begin{cases} 1, & \text{if } \sqrt{(x-0.25)^2 + (y-0.25)^2} \leq 0.15 \\ 0, & \text{otherwise} \end{cases}, \qquad \forall \mathbf{x} \in (0,1)^2, \tag{2b}$$

$$u(\mathbf{x}, t) = 0, \qquad \text{in the left and bottom boundaries,} \tag{2c}$$

where $\mathbf{a} = [1, 1]^T$. Our objective is to obtain the solution at $t = 0.5$. Considering the lumped mass matrix and forward Euler, the Galerkin solution at $t = t^{n+1}$ is given by $u_h(\mathbf{x}, t^{n+1}) = \sum_j U_j^{n+1} \varphi(\mathbf{x})$. The coefficients U_j^{n+1} are obtained from

$$m_i \left(\frac{U_i^{n+1} - U_i^n}{\Delta t} \right) + \sum_j \mathbf{a} \cdot \mathbf{c}_{ij} U_j^n = 0, \tag{3}$$

where $\mathbf{c}_{ij} = \int_\Omega \nabla \varphi_j(\mathbf{x}) \phi_i(\mathbf{x}) d\mathbf{x} \in \mathbb{R}^2$. Proceed as follows:

* Obtain the initial condition $u_h(\mathbf{x}, 0)$ via

  $$u_h(\mathbf{x}, 0) = \sum_j U_j^0 \varphi(\mathbf{x}),$$

  where $U_i^0 = u(\mathbf{x}_i, 0)$.

* Obtain (pre-compute) the entries of the lumped mass matrix and the entries $\mathbf{c}_{ij}$. Solve equation (3) at every time step up to $t = 0.5$. Note: for this problem use $\Delta t = 0.1 \Delta x$. Experiment with the value Δx.

Exercise 3: solving the 2D advection equation via a first-order method

Consider again the 2D advection equation (2). This time, we will use the following first-order discretization

$$m_i \left(\frac{U_i^{n+1} - U_i^n}{\Delta t} \right) + \sum_j \mathbf{a} \cdot \mathbf{c}_{ij} U_j^n - \sum_j d_{ij} U_j^n = 0, \tag{4}$$

where

$$d_{ij} = \begin{cases} \max(\mathbf{a} \cdot \mathbf{c}_{ij}, 0, \mathbf{a} \cdot \mathbf{c}_{ji}), & \text{if } j \neq i, \\ -\sum_{j \neq i} d_{ij} & \text{if } j = i. \end{cases}$$

Proceed as follows:

* Obtain the initial condition $u_h(\mathbf{x}, 0)$ via

  $$u_h(\mathbf{x}, 0) = \sum_j U_j^0 \varphi(\mathbf{x}),$$

  where $U_i^0 = u(\mathbf{x}_i, 0)$.

* Obtain (pre-compute) the entries of the lumped mass matrix and the entries $\mathbf{c}_{ij}$ and d_{ij}. Solve equation (4) at every time step to obtain $u_h(\mathbf{x}, t)$ at $t = 0.5$. Note: for this problem use $\Delta t = 0.1 \Delta x$. Experiment with the value Δx.

Exercise 4: solving the 2D advection equation via the Flux Corrected Transport

Following the slides from the course, the high-order method (3) can be written as

$$U_i^{H,n+1} = U_i^{L,n+1} + \frac{1}{m_i}\sum_j f_{ij} \tag{5}$$

where $u^{L,n+1}$ is the low-order solution (4) and $f_{ij} = -\Delta t d_{ij}(U_j^n - U_i^n)$ is an anti-diffusive flux. The goal of this problem is to apply flux limiters to f_{ij} to guarantee $U^{n+1} \in [0,1]$. The FCT method is given by

$$U_i^{n+1} = U_i^{L,n+1} + \frac{1}{m_i}\sum_j \alpha_{ij} f_{ij} \tag{6}$$

where α_{ij} are flux limiters computed as follows:

$$\begin{aligned}
R_i^+ &= \begin{cases} \min\left(1, \frac{Q_i^+}{P_i^+}\right), & \text{if } P_i^+ \neq 0, \\ 1 & \text{otherwise} \end{cases} & \qquad R_i^- &= \begin{cases} \min\left(1, \frac{Q_i^-}{P_i^-}\right), & \text{if } P_i^- \neq 0, \\ 1 & \text{otherwise} \end{cases}, \\
Q_i^+ &= m_i(U^{\max} - U_i^L), & Q_i^- &= m_i(U^{\min} - U_i^L), \\
P_i^+ &= \sum_j \max(0, f_{ij}), & P_i^- &= \sum_j \min(0, f_{ij}),
\end{aligned}$$

where $U^{\min} = 0$ and $U^{\max} = 1$. Proceed as follows:

* Obtain the initial condition $u_h(\mathbf{x}, 0)$ via

  $$u_h(\mathbf{x}, 0) = \sum_j U_j^0 \varphi(\mathbf{x}),$$

  where $U_i^0 = u(\mathbf{x}_i, 0)$.

* For every time step compute U_i^L, the anti-diffusive fluxes f_{ij}, and the flux limiters α_{ij}. From (6), obtain the solution U^{n+1}. Proceed until the final time $t = 0.5$. Note: for this problem use $\Delta t = 0.1\Delta x$. Experiment with the value Δx.

<u>*Hint 1:*</u> Verify that $0 \leq U_i^L \leq 1$, up to machine precision; that is, $\min(U_i^L)$ might be a small negative number, on the order of machine precision. Similarly, $\max(U_i^L)$ might be slightly larger than 1, on the order of machine precision. Once this is verified, do $U_i^L = \max(0, \min(1, U_i^L))$.

<u>*Hint 2:*</u> the code to implement the FCT method is the same for 1D or 2D problems. Therefore, you can use the 1D Matlab code in the repository.