# TAM 470 / CSE 450

# Homework 6

## Problem 1 (20 points)

**Part 1: PrairieLearn (16 points):**

Consider the following ODE (this ODE is based off of Moin Problem 4.26 and can represent the temperature distribution in a body whose cross-sectional area varies with position, but whose temperature variation in the lateral directions is negligible compared to the axial direction):

$$\frac{d^2T}{dx^2} + A(x)\frac{dT}{dx} + B(x)T = f(x) \quad \text{for} \quad x \in [0, L] \tag{1}$$

with

$$A(x) = -\frac{x+3}{x+1}, \quad B(x) = \frac{x+3}{(x+1)^2}, \quad \text{and} \quad f(x) = 2(x+1) + 3B(x) \tag{2}$$

Recall that using the direct method with $O(h^2)$ central difference schemes for each derivative term leads to a system of equations of the form:

$$\alpha_j y_{j+1} + \beta_j y_j + \gamma_j y_{j-1} = f_j, \quad j = 1, 2, ..., N-1 \tag{3}$$

The expressions for $\alpha_j$, $\beta_j$, and $\gamma_j$ were derived in class (you do not need to repeat these derivations).

Consider the following mixed boundary conditions:

$$a_0 y(0) + b_0 y'(0) = c_0 \tag{4}$$

$$a_L y(L) + b_L y'(L) = c_L \tag{5}$$

Suppose we wish to approximate the derivative terms in Equations 4 and 5 using the following one-sided difference schemes:

$$y'(0) = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2) \tag{6}$$

and

$$y'(L) = \frac{y_{N-2} - 4y_{N-1} + 3y_N}{2h} + O(h^2) \tag{7}$$

Go to PrairieLearn to write a function that numerically solves the ODE described in Equations 1–2 using the mixed boundary conditions given in Equations 4–5 with the difference schemes given in Equations 6–7.

Hint: In class, we covered how to handle mixed boundary conditions using the $O(h)$ forwards and backwards difference schemes. Follow the procedure outlined there, but using the $O(h)$ schemes given in Equations 6 and 7 instead. You can expect to need to modify the first and last rows of the coefficients matrix and the right-hand side vector from Equation 3, relative to the case of Dirichlet boundary conditions at both ends.

**Part 2 (4 points):**

Use the code you wrote in Part 1 to predict the temperature at the right end of the domain with length $L = 2$ if the left end is held fixed at a temperature of $T_0 = 4$ and the right end experiences the mixed boundary condition $T(L)+2T'(L) = 5$. Your solution should include the following:

1. (2 pts) Plot the solution for several values of $h$ to demonstrate convergence of the temperature prediction at the right end.
2. (2 pts) Compute a very accurate solution using a fine grid that we'll refer to as the "exact" solution. Make a plot of the error, $\tau = |T(L)_{exact} - T(L)_{numerical}|$ in the temperature solution at the right end of the domain for different values of $h$ to demonstrate second order convergence of the solution at this location (this confirms correct implementation of the boundary condition in your code). Be sure to show/explain why your plot demonstrates second order convergence of the error.

**Be sure to submit a screenshot of your PrairieLearn solution code and the code used to produce the plots**.

## Problem 2 (20 points)

*No coding required*

Consider the 1D transient heat equation for the temperature $T(x,t)$:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{8}$$

Suppose we wish to approximate the spatial derivative with the fourth-order central difference scheme (assume uniform grid spacing and $\alpha$ is constant throughout the domain):

$$\frac{\partial^2 T}{\partial x^2} = \frac{-T_{j-2} + 16T_{j-1} - 30T_j + 16T_{j+1} - T_{j+2}}{12\Delta x^2} + O(\Delta x^4) \tag{9}$$

(**a**) (10 pts) Use von Neumann stability analysis to show that the maximum allowable time step using forward Euler is $\Delta t_{max} = \frac{3}{8}\frac{\Delta x^2}{\alpha}$ when using this scheme for the spatial derivative.

(**b**) (10 pts) Repeat part (a), but instead using modified wavenumber analysis.

## Problem 3 (20 points)

*No coding required*

The 1D convection-diffusion equation (with convection in the $+x$ direction) is

$$\frac{\partial T}{\partial t} + c\frac{\partial T}{\partial x} = \alpha \frac{\partial^2 T}{\partial x^2} \quad \text{for } x \in [0, L], \quad t \geq 0 \tag{10}$$

(**a**) (10 pts) Show that for homogeneous boundary conditions $T(0,t) = T(L,t) = 0$, semi-discretization using second order central difference schemes for the spatial derivatives leads to a system of ODEs of the form $\frac{d\mathbf{T}}{dt} = \mathbf{AT}$, where $\mathbf{A}$ is a banded tridiagonal matrix:

$$\mathbf{A} = B\left[\frac{c}{2\Delta x} + \frac{\alpha}{\Delta x^2}, -2\frac{\alpha}{\Delta x^2}, -\frac{c}{2\Delta x} + \frac{\alpha}{\Delta x^2}\right] \tag{11}$$

(**b**) (5 pts) The eigenvalues $\lambda_j$ of $\mathbf{A}$ can be shown to be:

$$\lambda_j = -2\frac{\alpha}{\Delta x^2} + 2\sqrt{\left(\frac{\alpha}{\Delta x^2}\right)^2 - \left(\frac{c}{2\Delta x}\right)^2}\cos\frac{\pi j}{N}, \quad j = 1, 2, ..., N-1 \tag{12}$$

For $L = 1$, $N = 50$ ($\Delta x = \frac{L}{N}$), $\alpha = 0.001$, and $c = 0.08$, state whether the **forward Euler** scheme for time integration is unstable or conditionally stable; if conditionally stable, what is the maximum allowable value of $\Delta t$ for stability?

(**c**) (5 pts) For the same $L$, $N$, $\alpha$, and $c$ values from part (b), state whether the **leapfrog** scheme for time integration is unstable or conditionally stable; if conditionally stable, what is the maximum allowable value of $\Delta t$ for stability?

## Problem 4 (20 points)

(**a**) (16 pts) Go to PrairieLearn to write a function that solves the convection-diffusion PDE (Equation 10) using forward Euler for time integration for the following initial profile:

$$T(x,0) = \begin{cases} 1-(10x-1)^2 & \text{for } 0 \le x \le 0.2, \\ 0 & \text{for } 0.2 < x \le 1 \end{cases} \tag{13}$$

The boundary conditions are homogeneous: $T(0,t) = T(1,t) = 0$.

The exact solution to Equation 10 **for pure convection** (i.e. $\alpha = 0$) given the initial profile defined in Equation 13 is

$$T_{exact}(x,t) = \begin{cases} 1-[10(x-ct)-1]^2 & \text{for } 0 \le x-ct \le 0.2, \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

(**b**) (4 pts) Use your code from Problem 4 to produce a plot of the the exact solution with no diffusion (Equation 14) and the numerical solution **with** diffusion (use $\alpha = 0.001$, $N = 50$, $c = 0.08$, and time step defined by $\alpha\dfrac{\Delta t}{\Delta x^2} = 0.4$). Create the plots for $t = 4$ and $t = 8$, and include both the exact solution (no diffusion) and numerical solution (with diffusion) on the same axes in each figure (**two total plots**).

**Be sure to submit a screenshot of your PrairieLearn solution code and the code used to produce the plots.**

## Problem 5 (20 points)

Consider the 1D transient heat equation for a domain of length $L$:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \quad \text{for } x \in [0,L], \quad t \ge 0 \tag{15}$$

Suppose we wish to use this equation to model heat transfer in a metallic bar that is insulated along its sides. The bar begins at a uniform temperature $T_{hot}$. At time $t = 0$, the right end is exposed to air cooling via convection while the left end is maintained at a fixed temperature of $T_{hot}$. This situation corresponds to the following initial and boundary conditions:

$$T(x,0) = T_{hot} \quad \text{for } x \in (0,L] \tag{16}$$

$$T(0,t) = T_{hot}, \quad t \ge 0 \tag{17}$$

$$\frac{\partial T}{\partial x}(L,t) = \gamma(T_\infty - T(L,t)), \quad t \ge 0 \tag{18}$$

For the convection condition, $T_\infty$ represents the ambient air temperature and $\gamma$ represents the ratio of convection heat transfer coefficient to thermal conductivity, $\frac{h}{k}$.

(**a**) (14 pts) Go to PrairieLearn to implement the Crank-Nicolson scheme (trapezoid rule for time integration) for this problem. This scheme was derived in class and you do not need to repeat the derivation.

Your code should assume general parameters $T_{hot}$, $T_\infty$, $\alpha$, $\gamma$, $L$, and $t_{end}$, and define the space and time discretization using inputs `Nx` and `dt` to define space and time discretization, respectively. **Your code should use the first order difference scheme for the spatial derivative in the convection boundary condition**.

(**b**) Assume the following parameters: $L = 1.0$ m, $\alpha = 1.0 \times 10^{-4}$ m$^2$/s, $\gamma = 5$ m$^{-1}$, $T_{hot} = 100^o$ C, $T_\infty = 20^o$ C. Make the following plots (total of 3 plots)

- (2 pts) A plot of temperature $T$ vs position $x$ at times $t = 200, 1000, 2000$, and 4000 using $N_x = 20$ and $dt = 10$ (4 total curves on the same axes).
- (2 pts) A plot of temperature $T$ vs time $t$ at $x = L$ using $N_x = 100$ and time step values $dt = 100, 25, 10, 5, 1$ (5 total curves on the same axes).
- (2 pts) A plot of temperature $T$ vs time $t$ at $x = L$ using $dt = 1$ and $N_x = 20, 40, 100$ (3 total curves on the same axes).

**Be sure to submit a screenshot of your PrairieLearn solution code and the code used to produce the plots**.

## Problem 6: 4 credit-hour students only (20 points)

Consider the problem of hot fluid flowing in a channel with walls at $y = \pm 1$. The flow is in the $x$-direction. The incoming temperature is $T^0(y) = 1$. For $x > 0$, the wall temperature is 0. A schematic for the problem is begin below:

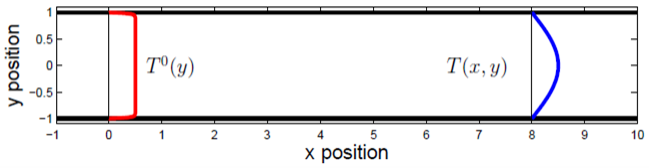


Figure 1: Schematic for the problem

The temperature for this problem is governed by the PDE

$$u(y)\frac{\partial T}{\partial x} = \alpha \frac{\partial^2 T}{\partial y^2}, \quad T(x=0,y) = T^0(y), \quad T(x, y \pm 1) = 0. \tag{19}$$

Here, the convecting velocity in the $x$-direction is $u(y) = 1 - y^2$, which is the velocity profile for laminar viscous flow.

In this problem, $x$ is the time-like variable and $y$ is the space variable. The problem is thus an initial boundary-value problem. It can be solved numerically with semi-discretization in $y$ and by applying a time integration scheme to $x$ (you simply 'time-march' the $x$ variable from 0 to $L$, where $L$ represents the end of the domain of interest).

The notation for representing the discretized solution in the domain is $T^i_j = T(x_i, y_j)$, where $i = 0, 1, 2, ..., N_x$ and $j = 0, 1, 2, ..., N_y$, such that there are $N_x + 1$ grid points in the $x$ direction and $N_y + 1$ grid points in the $y$ direction. Grid spacing parameters $\Delta x$ and $\Delta y$ do need to be the same.

(**a**) (3 pts) Use the second order central difference scheme for the $y$ discretization and the backward Euler scheme for the $x$ discretization to write down the solution update scheme at each "time step" in the form of a system of equations, $\mathbf{A}\mathbf{T}^{i+1} = \mathbf{T}^i$, where $\mathbf{T}^i = [T^i_1, T^i_2, ..., T^i_{N_y-1}]$ is the solution for the non-boundary nodes at position $x_i$. What is the form of the matrix $\mathbf{A}$?

(**b**) (3 pts) Use the second order central difference scheme for the $y$ discretization and the BDF2 scheme for the $x$ discretization to write down the solution update scheme at each "time step" in the form of a system of equations, $\mathbf{B}\mathbf{T}^{i+1} = c_1\mathbf{T}^i + c_2\mathbf{T}^{i-1}$, where $\mathbf{T}^i = [T^i_1, T^i_2, ..., T^i_{N_y-1}]$ is the solution for the non-boundary nodes at position $x_i$. What is the form of the matrix $\mathbf{B}$, and what are the coefficients $c_1$ and $c_2$?

(**c**) (10 pts) Go to Go to PrairieLearn to implement the BDF2 scheme to numerically solve Equation 19. Use backward Euler for the first step, then BDF2 for the remaining steps.

(**d**) (2 pts) Use your PrairieLearn solution code to solve for $T(x, y)$ on the interval $[0, 100] \times [-1, 1]$ with $\alpha = 0.01$. Demonstrate convergence of the solution by plotting the numerical solutions for $T(10, y)$ and $T(x, 0)$ for several values of $\Delta x$ and $\Delta y$ (Fix $\Delta x$ while varying $\Delta y$, and vice versa when varying $\Delta x$, to isolate the error influence of each parameter). Based on these plots, state the grid spacing values $\Delta x$ and $\Delta y$ that you feel achieve convergence.

(**e**) (2 pts) Produce a color contour plot of the converged solution on $[0, 100] \times [-1, 1]$. Suggested plotting syntax is given below, assuming you store a list of grid points in variables `x` and `y` and solution in a 2D array `T` and use `matplotlib`:

```
X, Y = np.meshgrid(x, y) # x and y are 1D arrays of grid points
plt.figure()
contour_regions = np.linspace(0, 1, 6)
plt.contourf(X, Y, T, levels=contour_regions, cmap='rainbow')
plt.xlabel('x')
plt.ylabel('y')
cbar = plt.colorbar()
cbar.set_label('T', rotation=0)
plt.grid()
plt.show()
```

**Be sure to submit a screenshot of your PrairieLearn solution code and the code used to produce the any plots**.