Advanced Numerical Methods for PDEs

Homework 1

Francisco Castillo

February 17, 2021

Consider the initial - boundary value problem for the scalar advection diffusion equation

$$\partial_t u(x,t) + a\partial_x u(x,t) - b\partial_x^2 u(x,t) = 0, \qquad u(x,t=0) = u^I(x), \tag{1}$$

on the interval $x \in [-1,1]$ with periodic boundary conditions $u(x+2,t) = u(x,t), \forall x, t$. Consider the explicit difference method

$$U(x,t+\Delta t) = U(x,t) - \frac{a\Delta t}{2\Delta x}(T - T^{-1})U(x,t) + \frac{b\Delta t}{\Delta x^2}(T - 2 + T^{-1})U(x,t) \tag{2}$$

for the problem (1).

Problem 1

1. Derive an analytic expression for the solution $u(x,t)$ of problem (1) for a general initial function $u^I(x)$ and general stepsizes Δx, Δt, using Fourier transforms.

Solution: We will use the Fourier transform

$$\hat{u}(w,t) = \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} u(x,t) e^{-i\omega x} dx, \tag{3}$$

to turn our $1-D$ PDE into an ODE. Taking the Fourier transform of the PDE (1) we obtain

$$\partial_t \hat{u}(w,t) + iaw\hat{u}(w,t) + bw^2\hat{u}(w,t) = 0,$$

which can be manipulated into

$$\partial_t \hat{u}(w,t) = -(bw^2 + iaw)\hat{u}(w,t).$$

The previous equation has a simple analytical solution,

$$\hat{u}(w,t) = e^{-(bw^2+iaw)t}\hat{u}(w,0),$$

which we can conveniently rewrite as

$$\hat{u}(w,t) = e^{-b\omega^2 t}\hat{u}^I(w)e^{-iawt} \tag{4}$$

To obtain the solution, we will use the following property of Fourier transforms,

$$\mathcal{F}\left[f * g\right] = \mathcal{F}\left[f\right]\mathcal{F}\left[g\right], \tag{5}$$

where $*$ represents convolution. In our case, we have

$$\begin{aligned}\mathcal{F}\left[f\right](w,t) &= e^{-b\omega^2 t},\\ \mathcal{F}\left[g\right](w,t) &= \hat{u}^I(w)e^{-iawt}.\end{aligned}$$

By using the inverse Fourier transform on the previous equations we obtain

$$\begin{aligned}f(x,t) &= \mathcal{F}^{-1}\left[\mathcal{F}\left[f\right](w,t)\right](x,t) = \frac{e^{-x^2/4bt}}{\sqrt{2bt}}, \qquad (6)\\ g(x,t) &= \mathcal{F}^{-1}\left[\mathcal{F}\left[g\right](w,t)\right](x,t) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\hat{u}^I(w)e^{-iw(x-at)}dw\\ &= u^I(x-at).\end{aligned}$$

Hence, we have proved so far that

$$\hat{u}(w,t) = e^{-b\omega^2 t}\hat{u}^I(w)e^{-iawt} = \mathcal{F}\left[f\right](w,t)\mathcal{F}\left[g\right](w,t) = \mathcal{F}\left[f * g\right](w,t), \tag{7}$$

Then, we can finally obtain the solution to the problem,

$$\begin{aligned}u(x,t) &= \mathcal{F}^{-1}\left[\hat{u}(w,t)\right](x,t)\\ &= \mathcal{F}^{-1}\left[\mathcal{F}\left[f * g\right](w,t)\right](x,t)\\ &= \left[f * g\right](x,t).\end{aligned}$$

To conclude,

$$u(x,t) = \left[f * u^I(x-at)\right](x,t), \tag{8}$$

with f given by (6).

2. Derive an analytic expression for the solution $U(x,t)$ of problem (2) for a general initial function $u^I(x)$ and general stepsizes Δx, Δt, using Discrete Fourier transforms.

Solution: We will use the Discrete Fourier transform

$$\hat{u}(w_\nu, t_n) = \frac{\Delta x}{\sqrt{2\pi}} \sum_{j=-N}^{N} u(x_j, t_n) e^{-i\omega_\nu x_j}, \tag{9}$$

and its inverse,

$$u(x_j, t_n) = \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} \hat{u}(w_\nu, t_n) e^{ix_j w_\nu}, \tag{10}$$

where $x_j = j\Delta_x$, $t_n = n\Delta t$, $w_\nu = \nu \Delta w$ and $N\Delta x \Delta w = \pi$. It is simple to prove that

$$\begin{aligned}
TU(x_j, t_n) = U(x_{j+1}, t_n) &= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} \hat{u}(w_\nu, t_n) e^{-i(x_{j+1})w_\nu} \\
&= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} e^{-i\Delta x w_\nu} \hat{u}(w_\nu, t_n) e^{-ix_j w_\nu}, \\
T^{-1}U(x_j, t_n) = U(x_{j-1}, t_n) &= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} \hat{u}(w_\nu, t_n) e^{-i(x_{j-1})w_\nu} \\
&= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} e^{i\Delta x w_\nu} \hat{u}(w_\nu, t_n) e^{-ix_j w_\nu},
\end{aligned}$$

We can substitute into equation(2) and simplify to obtain,

$$\begin{aligned}
\hat{u}(w_\nu, t_{n+1}) &= \left[1 - \frac{a\Delta t}{2\Delta x}(e^{-i\Delta x w_\nu} - e^{i\Delta x w_\nu}) + \frac{b\Delta t}{\Delta x^2}(e^{-i\Delta x w_\nu} - 2 + e^{i\Delta x w_\nu})\right] \hat{u}(x_j, t_n) \\
&= \left[1 + i\frac{a\Delta t}{\Delta x} \sin(\Delta x w_\nu) - \frac{4b\Delta t}{\Delta x^2} \sin^2\left(\frac{\Delta x w_\nu}{2}\right)\right] \hat{u}(w_\nu, t_n). \qquad (11)
\end{aligned}$$

Define $g(\omega_\nu)$ as

$$g(\omega_\nu) = \left[1 + i\frac{a\Delta t}{\Delta x} \sin(\Delta x w_\nu) - \frac{4b\Delta t}{\Delta x^2} \sin^2\left(\frac{\Delta x w_\nu}{2}\right)\right]. \tag{12}$$

Then, $\hat{u}(w_\nu, t_{n+1}) = g(\omega_\nu)\hat{u}(w_\nu, t_n)$. We can now find the solution, in frequency domain, as a function of the initial condition.

$$\begin{aligned}
\hat{u}(w_\nu, t_n) &= g(\omega_\nu)\hat{u}(w_\nu, t_{n-1}) = g^2(\omega_\nu)\hat{u}(w_\nu, t_{n-2}) = g^3(\omega_\nu)\hat{u}(w_\nu, t_{n-3}), \\
&= \cdots = g^n(\omega_\nu)\hat{u}(w_\nu, 0), \\
&= g^n(\omega_\nu)\hat{u}^I(w_\nu).
\end{aligned}$$

We now use equation (10) to obtain the solution in space domain,

$$\begin{aligned} u(x_j, t_n) &= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} \hat{u}(w_\nu, t_n) e^{i x_j w_\nu} \\ &= \frac{\Delta w}{\sqrt{2\pi}} \sum_{\nu=-N}^{N} g^n(\omega_\nu) \hat{u}^I(w_\nu) e^{i x_j w_\nu}. \end{aligned} \tag{13}$$

To finish, it must be mentioned that the above solution is stable if and only if $|g(w)| \leq 1, \forall w \in [-\frac{\pi}{\Delta x}, \frac{\pi}{\Delta x}]$. This condition will produce the following conditions, CFL conditions, on the parameters:

$$a^2 \Delta t \leq 2b, \text{ and } 2b\Delta t \leq \Delta x^2. \tag{14}$$

Their derivation is detailed in the appendix.

Problem 2

1. Write a program to solve (1) for $0 < t < T$ using the discretization (2) for general values of $a, b, \Delta x, \Delta t$ and a general initial function $u^I(x)$.

Solution: Let $u_j^n = u(x_0 + j\Delta x, n\Delta t)$, with $x_0 = -1$. Then, we can rewrite equation (2) into

$$\begin{aligned} u_j^{n+1} &= u_j^n - \frac{a\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right) + \frac{b\Delta t}{\Delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right), \\ &= u_j^n - \frac{1}{2}ac\left(u_{j+1}^n - u_{j-1}^n\right) + \frac{bc}{\Delta x}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right), \end{aligned}$$

where $c = \frac{\Delta t}{\Delta x}$. Regrouping terms we obtain

$$\begin{aligned} u_j^{n+1} &= c\left(\frac{b}{\Delta x} + \frac{a}{2}\right)u_{j-1}^n + \left(1 - \frac{2bc}{\Delta x}\right)u_j^n + c\left(\frac{b}{\Delta x} - \frac{a}{2}\right)u_{j+1}^n, \\ &= Au_{j-1}^n + Bu_j^n + Cu_{j+1}^n, \end{aligned} \tag{15}$$

where $A = c\left(\frac{b}{\Delta x} + \frac{a}{2}\right)$, $B = \left(1 - \frac{2bc}{\Delta x}\right)$ and $C = c\left(\frac{b}{\Delta x} - \frac{a}{2}\right)$. The previous equation can be represented as a tridiagonal system,

$$\vec{u}^{n+1} = M\vec{u}^n, \tag{16}$$

where M is a tridiagonal matrix with A, B and C being its lower, main, and upper diagonal, respectively. Note that

$$\vec{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_j \\ \vdots \\ u_N \end{pmatrix}$$

We can advance in time by simply using (22). To implement the periodic boundary conditions we consider the end points x_0 and x_N in equation (20). At x_0:

$$\begin{aligned} u_0^{n+1} &= Au_{-1}^n + Bu_0^n + Cu_1^n, \\ &= Au_{N-1}^n + Bu_0^n + Cu_1^n, \end{aligned}$$

since $u_{-1} = u_{N-1}$. At x_0:

$$\begin{aligned} u_N^{n+1} &= Au_{N-1}^n + Bu_N^n + Cu_{N+1}^n, \\ &= Au_{N-1}^n + Bu_N^n + Cu_2^n, \end{aligned}$$

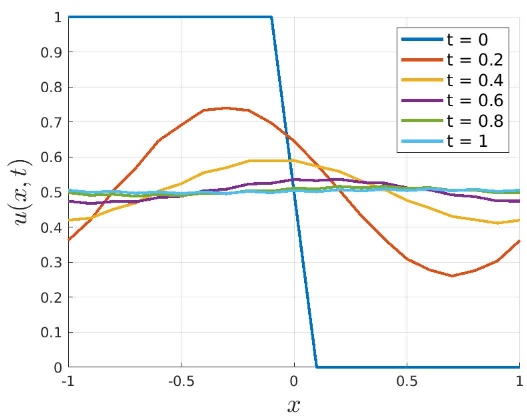
since $u_{N+1} = u_2$. This method was coded in Matlab (code at the end of this problem) and used to solve the next question.

2. Solve the discretized problem (2) for $0 < t < 1$, using the values $a = 1, b = 0.5$ and

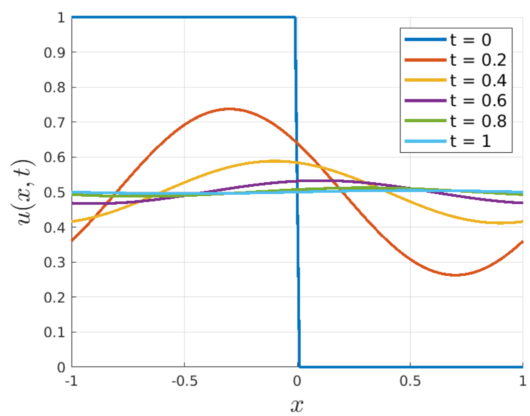
$$u^I(x) = \begin{cases} 1 & x \le 0 \\ 0 & x > 0 \end{cases} \tag{17}$$

Use stepsizes $\Delta x = 0.1$, $\Delta x = 0.01$ and $\Delta x = 0.001$. Use the analysis of Problem 1 to determine an appropriate stepsize Δt.

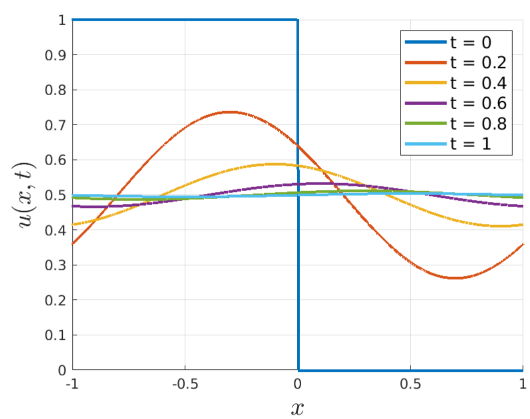
Solution: The method above was implemented. See figure 1 for the solution profiles at different times and using different mesh sizes. We can barely see any difference between the solutions for $\Delta x = 0.01$ (b) and $\Delta x = 0.001$ (c). Since the compute time is considerably larger for $\Delta x = 0.001$, with very little gain in accuracy, we don't see the need for such a fine mesh. At the same time, it can be observed that $\Delta x = 0.1$ is too big.



(a) $\Delta x = 0.1$.



(b) $\Delta x = 0.01$.



(c) $\Delta x = 0.001$.

Figure 1: Solution $u(x, t)$ against x for the PDE in (1) for different values of t and $a = 1$, $b = 0.5$.

Find the code that produced the plots in figure 1 below. The reader is welcome to set enableVideo = true, to see the evolution of the signal in real time. Further, if b were to be set to a value very close to zero, we would see a travelling wave that doesn't suffer any diffusion, as expected.To conclude, Δt has been chosen following the *CFL* condition given by (14), implemented in the function calculate_dt.

Matlab code:

```
clear all; close all; clc
format long

enableVideo = false;

a = 1.0;
b = 1/2;
dx = 1/100;
dt_default = calculate_dt(a,b,dx);
dt = dt_default;

x0 = -1;
xN = 1;
N = (xN-x0)/dx;
x = linspace(x0,xN,N+1)';
u0 = heaviside(-x);

[A,B,C] = calculateDiagonals(a,b,dx,dt);
Mtilde_default = calculate_Mtilde(A,B,C,N);
Mtilde = Mtilde_default;

t = 0;
u = u0;
T = 1;

k = 1:5;
plotTimes = k*T/5
% plotTimes = [.1,.2,.5,T]
storeCounter = 1;
shouldStore = false;
storedSolutions = [];
dtHasChanged = false;

while t<T
    if (dtHasChanged) % Need to reset values
        dt = dt_default;
        [A,B,C] = calculateDiagonals(a,b,dx,dt); % Lower Diag, Diag, Upper Diag
        Mtilde = Mtilde_default;
        dtHasChanged = false;
    end
    if(t+dt > plotTimes(storeCounter))
        dt = plotTimes(storeCounter) - t;
        % We need to recalculate the matrix for the new dt
```

```
        [A,B,C] = calculateDiagonals(a,b,dx,dt);
        Mtilde = calculate_Mtilde(A,B,C,N);
        shouldStore = true; % Should plot the solution after this iteration
        dtHasChanged = true;
    end

    % -- Advance solution --
    u_prev = u;
    u(2:N) = Mtilde * u_prev; % Solve the interior
    % Periodic BCs
    u(1)   = A*u_prev(N) + B*u_prev(1) + C*u_prev(2);
    u(N+1) = A*u_prev(N) + B*u_prev(N+1) + C*u_prev(2);

    % -- Advance time --
    t = t + dt;

    if(shouldStore)
        disp(['Storing solution at t = ',num2str(t)])
        storedSolutions = [storedSolutions u];
        storeCounter = storeCounter +1;
        shouldStore = false;
    end
    if(enableVideo)
        figure(1)
        grid on
        plot(x,u);
        axis([-1 1 min(u0) max(u0)])
    end
end

figName = create_figName(b,dx);
plot_solutions(x,[0 plotTimes],[u0 storedSolutions],[-1 1 min(u0) max(u0)],figName)

function figName = create_figName(b,dx)
    figName = 'sol_b';
    if (b==0)
        figName = append(figName,'0_dx');
    else
        exponent = floor(log10(b));
        base = b/10^(exponent);
        figName = append(figName,num2str(base),'e',num2str(exponent),'_dx');
    end
    exponent = floor(log10(dx));
    base = dx/10^(exponent);
    figName = append(figName,num2str(base),'e',num2str(exponent));
end

function Mtilde = calculate_Mtilde(A,B,C,N)
    M = diag(A*ones(1,N),-1) + diag(B*ones(1,N+1)) + diag(C*ones(1,N
```

```
),1);
    Mtilde = M(2:N,:); % For the interior
end

function [A,B,C] = calculateDiagonals(a,b,dx,dt)
    c = dt/dx; % Courant Number
    A = c*(b/dx + a/2); % Lower diagonal
    B = 1 - 2*b*c/dx;    % Main diagonal
    C = c*(b/dx - a/2); % Upper diagonal
end

function plot_solutions(x,times,solutions,axisLimits,figName)
    linewidth = 2;
    labelfontsize = 18;
    legendfontsize = 12;

    figure(2)
    grid on
    hold on
    for i=1:length(times)
        plot(x,solutions(:,i),'DisplayName',['t = ',num2str(times(i)
)],'linewidth',linewidth);
    end
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize
)
    l = legend;
    set(l,'fontsize',legendfontsize)
    axis(axisLimits)
    saveas(gcf,figName,'png')
end

function round_number = round_down(number, decimals)
    multiplier = 10^decimals;
    round_number = floor(number * multiplier)/multiplier;
end

function dt = calculate_dt(a,b,dx)
    dt = round_down(min(2*b/a^2, dx^2/(2*b)), 6);
end
```

Problem 3

1. Show that for $b = 0$ the exact solution of (1) is given by $u(x,t) = u^I(xat)$.

Solution: We retake (4), with $b = 0$,

$$\begin{aligned}\hat{u}(w,t) &= e^{-\cancel{b}\omega^2 t}\hat{u}^I(w)e^{-iawt},\\ &= \hat{u}^I(w)e^{-iawt}.\end{aligned}$$

We then find the solution using the inverse Fourier transform,

$$\begin{aligned}u(x,t) &= \int_{-\infty}^{\infty} \hat{u}(w,t)e^{iwx}dw,\\ &= \int_{-\infty}^{\infty} \hat{u}^I(w)e^{-iawt}e^{iwx}dw,\\ &= \int_{-\infty}^{\infty} \hat{u}^I(w)e^{iw(x-at)}dw,\\ &= u^I(x-at).\end{aligned}$$

It can also be proved by simplty substituting $u^I(x-at)$ into the PDE (with $b = 0$).

2. Use the program of Problem 2 to solve the equation

$$\partial_t u + a\partial_x u = 0, \tag{18}$$

with $a = 0.5$, in $t \in [0,4]$, $x \in [-1,1]$ with periodic boundary conditions and the initial function $u^I(x)$ from (17). Again, use $\Delta x = 0.1$, $\Delta x = 0.01$, $\Delta x = 0.001$.

Solution: The program for problem 2 cannot be used in this problem because $b = 0$. This causes M to be the identity and, more importantly, the CFL conditions cannot be met. We will adapt this problem using the Lax-Friedrichs scheme, which introduces an artificial diffusion to the problem. We will modify equation (2), with $b = 0$, a bit for this purpose:

$$U(x,t+\Delta t) = \frac{1}{2}\left(T + T^{-1}\right)U(x,t) - \frac{a\Delta t}{2\Delta x}(T - T^{-1})U(x,t) \tag{19}$$

Not that we have substituted $U(x,t)$ for $\frac{1}{2}\left(T+T^{-1}\right)U(x,t)$. Then, we can rewrite equation (19) into

$$\begin{aligned}u_j^{n+1} &= \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{a\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right),\\ &= \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{1}{2}ac\left(u_{j+1}^n - u_{j-1}^n\right).\end{aligned}$$

where $c = \frac{\Delta t}{\Delta x}$, as before. Regrouping terms we obtain

$$\begin{aligned} u_j^{n+1} &= \frac{1}{2}\left(1+ac\right)u_{j-1}^n + \frac{1}{2}\left(1-ac\right)u_{j+1}^n, \\ &= A'u_{j-1}^n + B'u_j^n + C'u_{j+1}^n, \end{aligned} \tag{20}$$

where $A' = \frac{1}{2}\left(1+ac\right)$, $B' = 0$ and $C' = \frac{1}{2}\left(1-ac\right)$. The previous equation can be represented as a tridiagonal system,

$$\vec{u}^{n+1} = M'\vec{u}^n, \tag{21}$$

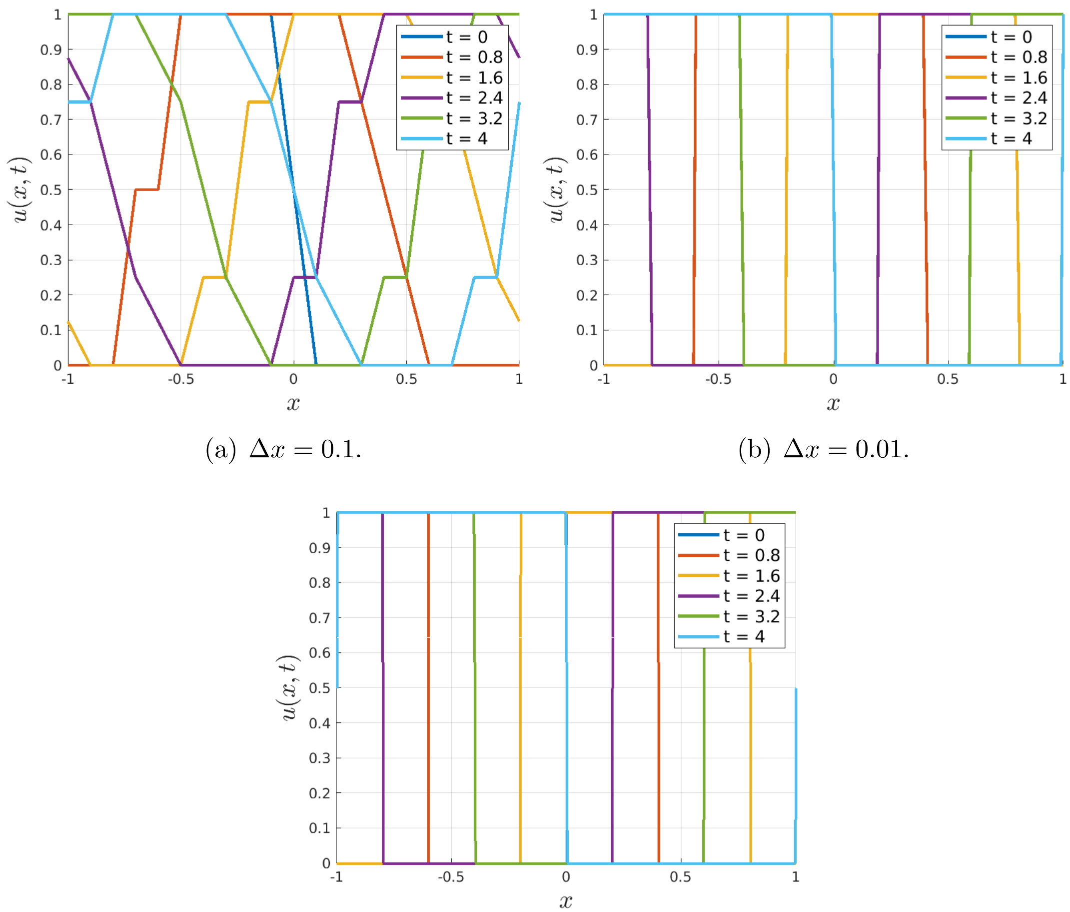
where M' is a tridiagonal matrix with A', B' and C' being its lower, main, and upper diagonal, respectively. We close the problem by implementing the boundary conditions in the same manner as in Problem 2, but with the new values A', B', C'. The new CFL condition, derived in the appendix, is

$$\Delta t \leq \frac{\Delta x}{|a|}.$$

Hence, to implement this new method, it sufices with modifying the code from Problem 2. We will check the value of b and, if zero, we will define A, B, C with the new values just presented. In addition, the value of Δt will be also derived from the new CFL condition. In figure 3 we can see that, without diffusion (other than the negligible artificial diffusion introduced by the Lax-Friedrichs scheme) we obtain a travelling solution $u(x,t) = u^I(x - at)$.

As we can see, $\Delta x = 0.1$ is not good enough, as it doesn't capture the step function well enough. In this case, we can appreciate an improvement when using $\Delta x = 0.001$ vs $\Delta x = 0.01$, and the simulation doesn't take that much longer. In fact, the all simulations take considerably less time than when $b \neq 0$ like in the previous problem. We can then raise the conclusion that most of the compute time is spent on the diffussion term. It is somewhat difficult to see the solutions at different times t, since they are supperposed because of the lack of diffusion. We recommend enabling video to see the step function move with time. Note that, because $a = 0.5$ the wave travels 0.5 units in x every unit of time t.

Please find the Matlab code below figure 2.



(a) $\Delta x = 0.1$.

(b) $\Delta x = 0.01$.

(c) $\Delta x = 0.001$.

Figure 2: Solution $u(x,t)$ against x for the PDE in (18) for different values of t and $a = 0.5$.

Matlab code:

```
clear all; close all; clc
format long

enableVideo = false;

a = 0.5;
b = 0;
dx = 0.1;
dt_default = calculate_dt(a,b,dx);
dt = dt_default;

x0 = -1;
xN = 1;
N = (xN-x0)/dx;
x = linspace(x0,xN,N+1)';
```

```
u0 = heaviside(-x);

[A,B,C] = calculateDiagonals(a,b,dx,dt);
Mtilde_default = calculate_Mtilde(A,B,C,N);
Mtilde = Mtilde_default;

t = 0;
u = u0;
T = 4;

k = 1:5;
plotTimes = k*T/5

storeCounter = 1;
shouldStore = false;
storedSolutions = [];
dtHasChanged = false;

while t<T
    if (dtHasChanged) % Need to reset values
        dt = dt_default;
        [A,B,C] = calculateDiagonals(a,b,dx,dt);
        Mtilde = Mtilde_default;
        dtHasChanged = false;
    end
    if(t+dt > plotTimes(storeCounter))
        dt = plotTimes(storeCounter) - t;
        % We need to recalculate the matrix for the new dt
        [A,B,C] = calculateDiagonals(a,b,dx,dt);
        Mtilde = calculate_Mtilde(A,B,C,N);
        shouldStore = true; % Should plot the solution after this
   iteration
        dtHasChanged = true;
    end

    % -- Advance solution --
    u_prev = u;
    u(2:N) = Mtilde * u_prev; % Solve the interior
    % Periodic BCs
    u(1)   = A*u_prev(N) + B*u_prev(1) + C*u_prev(2);
    u(N+1) = A*u_prev(N) + B*u_prev(N+1) + C*u_prev(2);

    % -- Advance time --
    t = t + dt;

    if(shouldStore)
        disp(['Storing solution at t = ',num2str(t)])
        storedSolutions = [storedSolutions u];
        storeCounter = storeCounter +1;
        shouldStore = false;
    end
    if(enableVideo)
```

```matlab
        figure(1)
        grid on
        plot(x,u);
        axis([-1 1 min(u0) max(u0)])
    end
end

figName = create_figName(b,dx);
plot_solutions(x,[0 plotTimes],[u0 storedSolutions],[-1 1 min(u0)
    max(u0)],figName)

function figName = create_figName(b,dx)
    figName = 'sol_b';
    if (b==0)
        figName = append(figName,'0_dx');
    else
        exponent = floor(log10(b));
        base = b/10^(exponent);
        figName = append(figName,num2str(base),'e',num2str(exponent)
    ,'_dx');
    end
    exponent = floor(log10(dx));
    base = dx/10^(exponent);
    figName = append(figName,num2str(base),'e',num2str(exponent));
end

function Mtilde = calculate_Mtilde(A,B,C,N)
    M = diag(A*ones(1,N),-1) + diag(B*ones(1,N+1)) + diag(C*ones(1,N
    ),1);
    Mtilde = M(2:N,:); % For the interior
end

function [A,B,C] = calculateDiagonals(a,b,dx,dt)
    c = dt/dx; % Courant Number
    if (b==0) % Lax-Friedrichs
        A = (1 + a*c)/2; % Lower diagonal
        B = 0;           % Main diagonal
        C = (1 - a*c)/2; % Upper diagonal
    else % FTCS
        A = c*(b/dx + a/2); % Lower diagonal
        B = 1 - 2*b*c/dx;   % Main diagonal
        C = c*(b/dx - a/2); % Upper diagonal
    end
end

function plot_solutions(x,times,solutions,axisLimits,figName)
    linewidth = 2;
    labelfontsize = 18;
    legendfontsize = 12;

    figure(2)
    grid on
```

```
    hold on
    for i=1:length(times)
        plot(x,solutions(:,i),'DisplayName',['t = ',num2str(times(i))],'linewidth',linewidth);
    end
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize)
    l = legend;
    set(l,'fontsize',legendfontsize)
    axis(axisLimits)
    saveas(gcf,figName,'png')
end

function round_number = round_down(number, decimals)
    multiplier = 10^decimals;
    round_number = floor(number * multiplier)/multiplier;
end

function dt = calculate_dt(a,b,dx)
    if (b==0) % Lax-Friedrichs
        dt = round_down(dx/abs(a), 6);
    else % FTCS
        dt = round_down(min(2*b/a^2, dx^2/(2*b)), 6);
    end
end
```

Problem 4

1. Repeat problem 3 using the Lax-Wendroff scheme.

Solution: As we did before, we are going to use matrix form for the Lax-Wendroff scheme. This is possible because in our PDE

$$\partial_t u(x,t) + \partial_x f(u(x,t)) = 0,$$

we have $f(u(x,t)) = au(x,t)$ and,

$$\partial_t u(x,t) + \partial_x f(u(x,t)) = \partial_t u(x,t) + a\partial_x u(x,t) = 0.$$

This is very important to "join" both steps and represent it in matrix form. For the first half-step, we use Lax-Friedrichs in a staggered mesh:

$$\begin{aligned} u_{j+1/2}^{n+1/2} &= \frac{1}{2}\left(u_{j+1}^n + u_j^n\right) - \frac{a\Delta t/2}{2\Delta x/2}\left(u_{j+1}^n - u_j^n\right), \\ &= \frac{1}{2}\left(1+ac\right)u_j^n + \frac{1}{2}\left(1-ac\right)u_{j+1}^n, \end{aligned}$$

where $c = \frac{\Delta t}{\Delta x}$. Moving to the second step, $FTCS$,

$$u_j^{n+1} = u_j^n - \frac{\Delta t/2}{\Delta x/2}\left(f\left(u_{j+1/2}^{n+1/2}\right) - f\left(u_{j-1/2}^{n+1/2}\right)\right).$$

Thanks to that $f(u(x,t)) = au(x,t)$, we can write:

$$\begin{aligned} u_j^{n+1} &= u_j^n - a\frac{\Delta t/2}{\Delta x/2}\left(u_{j+1/2}^{n+1/2} - u_{j-1/2}^{n+1/2}\right), \\ &= u_j^n - ac\left(u_{j+1/2}^{n+1/2} - u_{j-1/2}^{n+1/2}\right). \end{aligned}$$

Substituting the half grid values from the previous half-step, we get

$$\begin{aligned} u_j^{n+1} &= u_j^n - ac\left(u_{j+1/2}^{n+1/2} - u_{j-1/2}^{n+1/2}\right), \\ &= u_j^n - ac\left(\frac{1}{2}\left(1+ac\right)u_j^n + \frac{1}{2}\left(1-ac\right)u_{j+1}^n - \frac{1}{2}\left(1+ac\right)u_{j-1}^n - \frac{1}{2}\left(1-ac\right)u_j^n\right), \\ &= \frac{1}{2}ac\left(ac+1\right)u_{j-1}^n + \left(1-a^2c^2\right)u_j^n + \frac{1}{2}ac\left(ac-1\right)u_{j+1}^n, \\ &= A''u_{j-1}^n + B''u_j^n + C''u_{j+1}^n, \end{aligned}$$

where $A'' = \frac{1}{2}ac\left(ac+1\right)$, $B'' = 1-a^2c^2$ and $C' = \frac{1}{2}ac\left(ac-1\right)$. The previous equation can be represented as a tridiagonal system,

$$\vec{u}^{n+1} = M''\vec{u}^n, \tag{22}$$

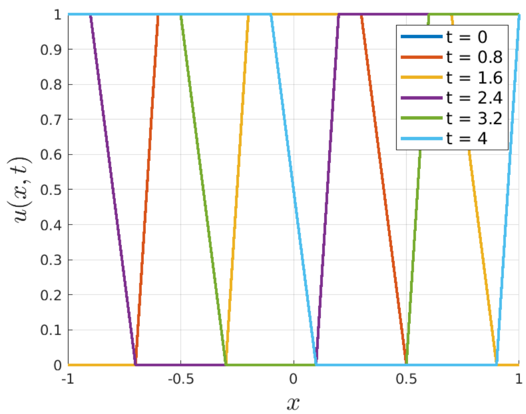
where M'' is a tridiagonal matrix with A'', B'' and C'' being its lower, main, and upper diagonal, respectively. We close the problem by implementing the boundary conditions in the same manner as in Problem 2, but with the new values A'', B'', C''. The new CFL condition, derived in the appendix, is

$$\Delta t \leq \frac{\Delta x}{|a|}.$$

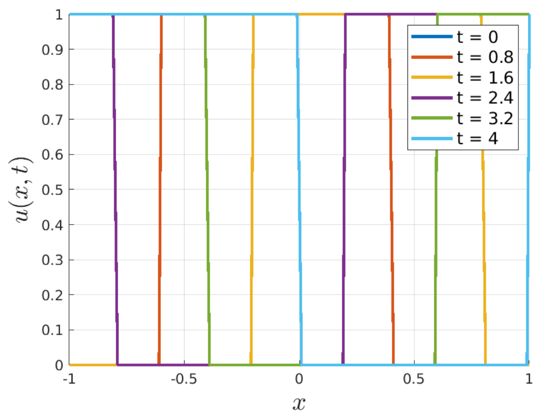
Please find the Matlab code below figure 3.

2. Discuss the difference to the Lax-Friedrichs solution.

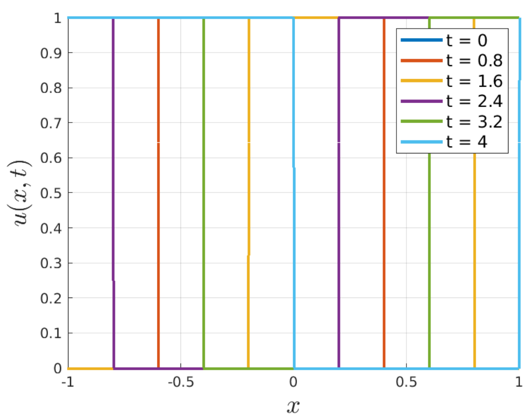
Solution: In figure 3 we can find the solution to the PDE (18), using Lax-Wendroff instead of Lax-Friedrichs. Compating figure 2 and 3, we can observe that for a coarser mesh, the Lax-Wendroff is far more accurate. In other words, the method converges faster. This is because the Lax-Wendroff method is a second order accurate both in space and time. On the other hand, Lax-Friedrichs is only first order accurate.



(a) $\Delta x = 0.1$.



(b) $\Delta x = 0.01$.



(c) $\Delta x = 0.001$.

Figure 3: Solution $u(x,t)$ against x for the PDE in (18) for different values of t and $a = 0.5$.

Matlab code:

```
clear all; close all; clc
format long

enableVideo = false;

a = 0.5;
b = 0;
dx = 0.001;
dt_default = calculate_dt(a,b,dx);
dt = dt_default;

x0 = -1;
xN = 1;
N = (xN-x0)/dx;
x = linspace(x0,xN,N+1)';
```

```
u0 = heaviside(-x);

[A,B,C] = calculateDiagonals(a,b,dx,dt);
Mtilde_default = calculate_Mtilde(A,B,C,N);
Mtilde = Mtilde_default;

t = 0;
u = u0;
T = 4;

k = 1:5;
plotTimes = k*T/5

storeCounter = 1;
shouldStore = false;
storedSolutions = [];
dtHasChanged = false;

while t<T
    if (dtHasChanged) % Need to reset values
        dt = dt_default;
        [A,B,C] = calculateDiagonals(a,b,dx,dt);
        Mtilde = Mtilde_default;
        dtHasChanged = false;
    end
    if(t+dt > plotTimes(storeCounter))
        dt = plotTimes(storeCounter) - t;
        % We need to recalculate the matrix for the new dt
        [A,B,C] = calculateDiagonals(a,b,dx,dt);
        Mtilde = calculate_Mtilde(A,B,C,N);
        shouldStore = true; % Should plot the solution after this iteration
        dtHasChanged = true;
    end

    % -- Advance solution --
    u_prev = u;
    u(2:N) = Mtilde * u_prev; % Solve the interior
    % Periodic BCs
    u(1)   = A*u_prev(N) + B*u_prev(1) + C*u_prev(2);
    u(N+1) = A*u_prev(N) + B*u_prev(N+1) + C*u_prev(2);

    % -- Advance time --
    t = t + dt;

    if(shouldStore)
        disp(['Storing solution at t = ',num2str(t)])
        storedSolutions = [storedSolutions u];
        storeCounter = storeCounter +1;
        shouldStore = false;
    end
    if(enableVideo)
```

```
        figure(1)
        grid on
        plot(x,u);
        axis([-1 1 min(u0) max(u0)])
    end
end

figName = create_figName(b,dx);
plot_solutions(x,[0 plotTimes],[u0 storedSolutions],[-1 1 min(u0) max(u0)],figName)

function figName = create_figName(b,dx)
    figName = 'sol_b';
    if (b==0)
        figName = append(figName,'0_dx');
    else
        exponent = floor(log10(b));
        base = b/10^(exponent);
        figName = append(figName,num2str(base),'e',num2str(exponent),'_dx');
    end
    exponent = floor(log10(dx));
    base = dx/10^(exponent);
    figName = append(figName,num2str(base),'e',num2str(exponent));
end

function Mtilde = calculate_Mtilde(A,B,C,N)
    M = diag(A*ones(1,N),-1) + diag(B*ones(1,N+1)) + diag(C*ones(1,N),1);
    Mtilde = M(2:N,:); % For the interior
end

function [A,B,C] = calculateDiagonals(a,b,dx,dt)
    c = dt/dx; % Courant Number
    if (b==0) % Lax-Wendroff
        A = 0.5*a*c*(a*c+1);
        B = 1-a^2*c^2;
        C = 0.5*a*c*(a*c-1);
    else % FTCS
        A = c*(b/dx + a/2); % Lower diagonal
        B = 1 - 2*b*c/dx;   % Main diagonal
        C = c*(b/dx - a/2); % Upper diagonal
    end
end

function plot_solutions(x,times,solutions,axisLimits,figName)
    linewidth = 2;
    labelfontsize = 18;
    legendfontsize = 12;

    figure(2)
    grid on
```

```
    hold on
    for i=1:length(times)
        plot(x,solutions(:,i),'DisplayName',['t = ',num2str(times(i)
    )],'linewidth',linewidth);
    end
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize
    )
    l = legend;
    set(l,'fontsize',legendfontsize)
    axis(axisLimits)
    saveas(gcf,figName,'png')
end

function round_number = round_down(number, decimals)
    multiplier = 10^decimals;
    round_number = floor(number * multiplier)/multiplier;
end

function dt = calculate_dt(a,b,dx)
    if (b==0) % Lax-Wendroff
        dt = round_down(dx/abs(a), 6);
    else % FTCS
        dt = round_down(min(2*b/a^2, dx^2/(2*b)), 6);
    end
end
```

Appendix

CFL conditions

Problem 1-2

For stability we need

$$
\begin{aligned}
|g(w)| &\leq 1, \\
\left| 1 + i\frac{a\Delta t}{\Delta x}\sin\left(\Delta x w_\nu\right) - \frac{4b\Delta t}{\Delta x^2}\sin^2\left(\frac{\Delta x w_\nu}{2}\right) \right| &\leq 1.
\end{aligned}
$$

We can use the identity $|g(w)|^2 = \Re\left[g\right]^2 + \Im\left[g\right]^2$ to obtain

$$
\left(1 - 4bcy^2\right)^2 + \left(2ac\Delta xy\sqrt{1-y^2}\right)^2 \leq 1,
$$

where c is the Courant Number and $y = \sin\left(w\Delta x/2\right) \in [-1, 1]$. We continue expanding the terms,

$$
\begin{aligned}
1 + 16b^2c^2y^4 - 8bcy^2 + 4a^2c^2\Delta x^2y^2(1-y^2) &\leq 1, \\
16b^2c^2y^4 - 8bcy^2 + 4a^2c^2\Delta x^2y^2(1-y^2) &\leq 0,
\end{aligned}
$$

Let $z = y^2$,

$$
\begin{aligned}
16b^2c^2z^2 - 8bcz + 4a^2c^2\Delta x^2z(1-z) &\leq 0, \\
16b^2c^2z - 8bc + 4a^2c^2\Delta x^2(1-z) &\leq 0,
\end{aligned}
$$

The previous equation represents a straight line on $z \in [0, 1]$. To guarantee that the entire line is negative, the endpoints must be.

- $z = 1$:

$$
\begin{aligned}
16b^2c^2 - 8bc &\leq 0, \\
2bc &\leq 1, \\
2b\frac{\Delta t}{\Delta x^2} &\leq 1, \\
2b\Delta t &\leq \Delta x^2.
\end{aligned}
$$

- $z = 0$:

$$
\begin{aligned}
-8bc + 4a^2c^2\Delta x^2 &\leq 0, \\
a^2c\Delta x^2 &\leq 2b, \\
a^2\Delta t &\leq 2b.
\end{aligned}
$$

Thus, the CFL conditions are

$$
2b\Delta t \leq \Delta x^2, \tag{23}
$$

$$
a^2\Delta t \leq 2b. \tag{24}
$$

Problem 3

For this problem the we have introduced an artificial diffusion by modifying the discrete PDE when adopting the Lax-Friedrichs scheme. We can obtain the desired CFL condition by removing b from the 2 previous CFL conditions. From (23) and (24),

$$a^2 \Delta t \leq 2b,$$
$$2b \leq \frac{\Delta x^2}{\Delta t}.$$

Joining both together we get

$$a^2 \Delta t \leq 2b \leq \frac{\Delta x^2}{\Delta t},$$
$$a^2 \Delta t \leq \frac{\Delta x^2}{\Delta t},$$
$$\Delta t^2 \leq \frac{\Delta x^2}{a^2}.$$

Wehre we have used the fact that the artificial diffusion introduced is $2b = \Delta x^2/\Delta t$. Finally, we obtain the desired CFL condition when $b = 0$,

$$\Delta t \leq \left| \frac{\Delta x}{a} \right|.$$

Problem 4

For this problem the we have introduced an artificial diffusion by modifying the discrete PDE when adopting the Lax-Wendroff scheme. We can obtain the desired CFL condition by removing b from the 2 previous CFL conditions. From (23) and (24),

$$a^2 \Delta t \leq 2b,$$
$$2b \leq \frac{\Delta x^2}{\Delta t}.$$

Joining both together we get

$$a^2 \Delta t \leq 2b \leq \frac{\Delta x^2}{\Delta t},$$
$$a^2 \Delta t \leq \frac{\Delta x^2}{\Delta t},$$
$$\Delta t^2 \leq \frac{\Delta x^2}{a^2}.$$

Where we have used the fact that the artificial diffusion introduced is $2b = a^2 \Delta t$. Finally, we obtain the desired CFL condition when $b = 0$,

$$\Delta t \leq \left| \frac{\Delta x}{a} \right|.$$