Final for MATH5311

Yan Bokai
Department of Mathematics

December 20, 2022

Problem 1

1. Consider the transport equation, where we assume a is a constant:

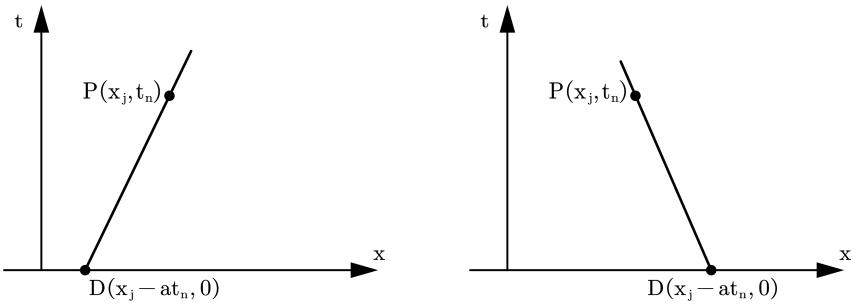
$$\begin{cases} u_t + au_x = 0, \quad t > 0 \\ u(x,0) = u_0(x) \end{cases}$$

The characteristic of the above equation is

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = a; \\ \frac{\mathrm{d}u}{\mathrm{d}t} = 0. \end{cases}$$

Thus the solution $u(x,t)$ is a constant along the characteristic $x - at = \xi$, where the characteristic intersects with the x-axis at $(\xi, 0)$.

At any point (x_j, t_n) , we can always find a characteristic passing through this point, and intersects with the x-axis at $(x_j - at_n, 0)$. This is the domain of dependence for the transport equation at point (x_j, t_n).



(a) Characteristic when $a > 0$. (b) Characteristic when $a < 0$.

Figure 1: Characteristic of the advection equation.

2. Consider the following scheme to solve the advection equation:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + a\frac{U_{j+2}^n - U_{j+1}^n}{\Delta x} = 0$$

According to the above numerical scheme, in order to calculate the value at point (x_j, t_n), we need information on the values of the other three points: (x_j, t_{n-1}), (x_{j+1}, t_{n-1}), (x_{j+2}, t_{n-1}). Recursively, the solution of every spatial point at fixed time step needs these kind of three points at previous time steps. Thus we can sketch the domain of dependence for this numerical scheme at point (x_j, t_n).

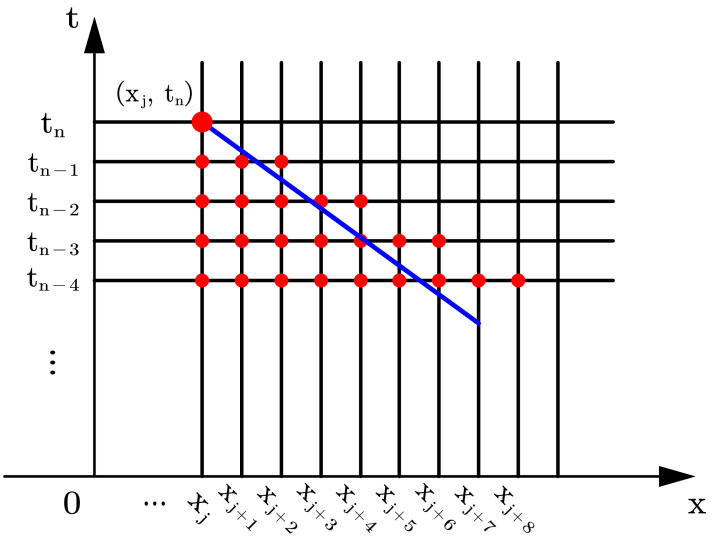


Figure 2: Domain of dependence.

3. CFL condition says that for a convergent scheme, the domain of dependence must lie within the domain of dependence of the numerical scheme.

   If $a > 0$, the domain of dependence of the equation is not contained in the domain of dependence of the numerical scheme, the CFL condition is not satisfied.

   If $a < 0$, in order to be consistent with CFL condition, we should set

$$\left| a \frac{\Delta t}{\Delta x} \right| = |v| \leq 2.$$

4. The CFL condition is a necessary condition for stability, so the CFL condition doesn't imply stability.

   Let $u_j^n = \lambda^n e^{ikj\Delta x}$, and plug it into the original equation. Then we have

$$\begin{aligned} \lambda &= 1 - v\left(e^{ik2\Delta x} - e^{ik\Delta x}\right), \\ |\lambda|^2 &= 4v^2 \sin^2\left(\frac{k\Delta x}{2}\right) - 2v[\cos(2k\Delta x) - \cos k(\Delta x)] + 1. \end{aligned}$$

   The scheme is stable if and only if $|\lambda| \leq 1$. But for any v, we can find k and Δx such that $|\lambda| > 1$, indicating the scheme is unconditionally unstable.

Problem 2

1. The Crank-Nicholson scheme is

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

2. If $j = N$, we assign $u_N^n = u(1, t_n) = 0$ for all $n \geq 0$.

   If $j = 0$, we introduce the ghost points $x_{-1} = -\Delta x$ and ghost value $u_{-1}^n = u(x_{-1}, t_n)$, then the scheme becomes

$$\frac{u_0^{n+1} - u_0^n}{\Delta t} = \frac{1}{2}\frac{u_1^n - 2u_0^n + u_{-1}^n}{(\Delta x)^2} + \frac{1}{2}\frac{u_1^{n+1} - 2u_0^{n+1} + u_{-1}^{n+1}}{(\Delta x)^2} \tag{1}$$

   Considering the boundary condition

$$\frac{\partial u}{\partial x} = \alpha(t)u + g(t) \text{ at } x = 0$$

   we have

$$\frac{u_1^n - u_{-1}^n}{2\Delta x} = \alpha(t_n)\, u_0^n + g(t_n).$$

   By eliminating u_{-1}^n and u_{-1}^{n+1}, (1) can be simplified to

$$\frac{u_0^{n+1} - u_0^n}{\Delta t} = \frac{u_1^n - u_0^n - \Delta x\left[\alpha(t_n)\, u_0^n + g(t_n)\right]}{(\Delta x)^2} + \frac{u_1^{n+1} - u_0^{n+1} - \Delta x\left[\alpha(t_{n+1})\, u_0^{n+1} + g(t_{n+1})\right]}{(\Delta x)^2}.$$

   Since we have N unknowns and N equations for a given n, we can solve them simultaneously.

3. Let $T_j^{n+\frac{1}{2}}$ be the truncation error. By definition, $T_j^{n+\frac{1}{2}}$ satisfies

$$\begin{aligned}\frac{u(x_j, t_{n+1}) - u(x_j, t_n)}{\Delta t} =& \frac{u(x_{j+1}, t_n) - 2u(x_j, t_n) + u(x_{j-1}, t_n)}{2(\Delta x)^2} \\ &+ \frac{u(x_{j+1}, t_{n+1}) - 2u(x_j, t_{n+1}) + u(x_{j-1}, t_{n+1})}{2(\Delta x)^2} + T_j^{n+\frac{1}{2}}\end{aligned}$$

   We expand all terms at $(x_j, t_{n+\frac{1}{2}})$ and by cancellation,

$$T_j^{n+\frac{1}{2}} = \left[\frac{1}{24}u_{ttt}(x_j, t_{n+\frac{1}{2}}) + \frac{1}{8}u_{xx}(x_j, t_{n+\frac{1}{2}})\right](\Delta t)^2 + \frac{1}{12}u_{xxxx}(x_j, t_{n+\frac{1}{2}})(\Delta x)^2 + o((\Delta t)^2) + o((\Delta x)^2).$$

   So the Crank-Nicholson scheme is second order in both time and space.

Problem 3

1. Assume the wave speed c is a constant. Let $u_k^n = \lambda^n e^{ijk\Delta x}$, and plug it into the leap-frog scheme, we have

$$\lambda - \frac{1}{\lambda} + 2ick\Delta t = 0.$$

   Then $\lambda = -ick\Delta t \pm \sqrt{1 - (ck\Delta t)^2}$. If $|ck\Delta t| \leq 1$, $|\lambda| = 1$. If $|ck\Delta t| > 1$, $|\lambda| = \left|ck\Delta t + \sqrt{(ck\Delta t)^2 - 1}\right| > 1$.

   Therefore, the scheme is stable when $\Delta t \leq 1/|cN|$.

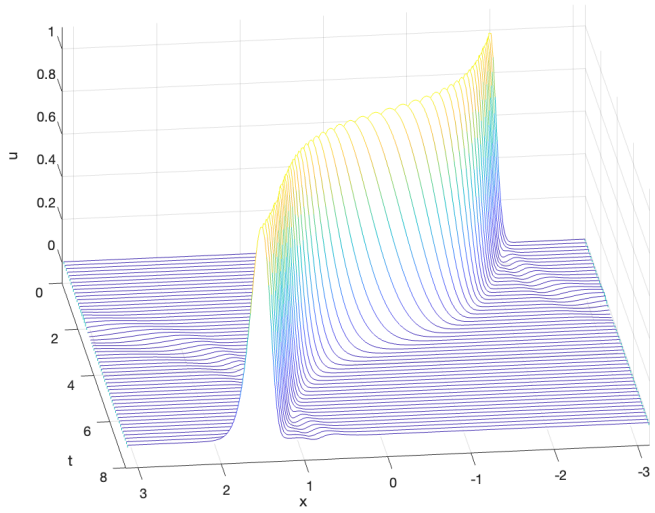


Figure 3: Propagation of the wave.

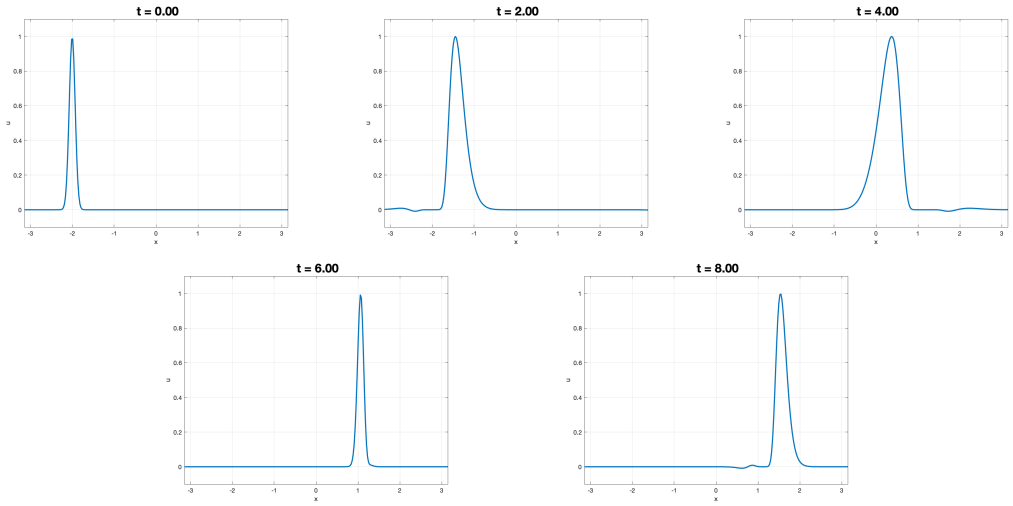


Figure 4: Propagation of the wave ($\Delta t = 0.005$).

2. If we choose $\Delta t = 0.005$, we find that the wave propagates along the positive direction of the x-axis, and the propagation speed is related to $c(x)$.

   If we choose $\Delta t = 0.0075$, oscillations will appear after a few iterations just because of a slight change in the value of Δt. Although the speed of the wave in the problem is not a constant, the stability of the scheme is still highly dependent on the selection of Δt.

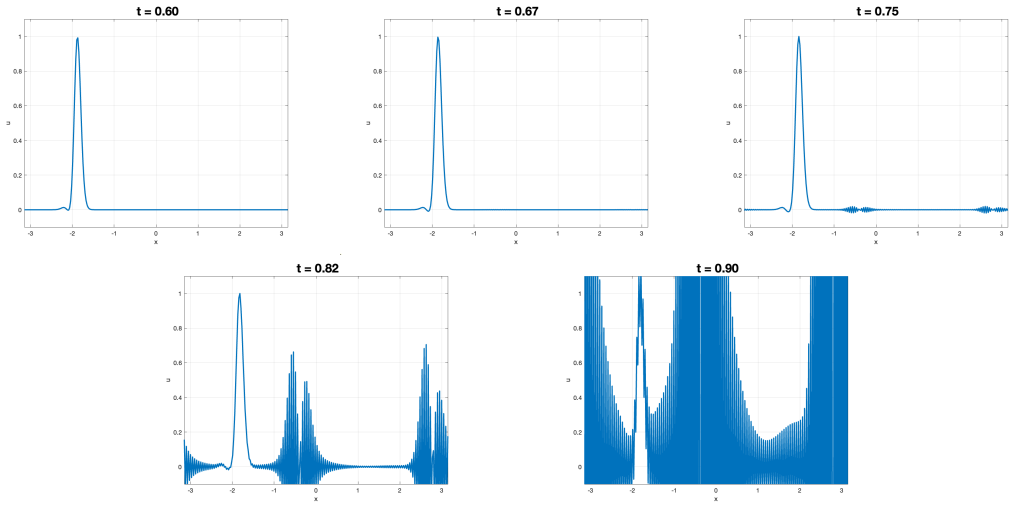


Figure 5: Propagation of the wave ($\Delta t = 0.0075$).

Problem 4

1. Let $k_x = 1, 2, \ldots, J$ and $k_y = 1, 2, \ldots, (J-1), (x_r, y_s) = (r\Delta x, s\Delta y)$. Assert the eigenvectors of A has the following form:

$$\{u_{r,s}^{k_x,k_y}\} = \left\{\cos\left(k_x - \frac{1}{2}\right)\pi x_r \sin k_y \pi y_s\right\}$$

As for the Jacobi iterative matrix G, let $A = D - L - U$, we have

$$G = D^{-1}(L+U).$$

Then we want to proof $\{u_{r,s}^{k_x,k_y}\}$ are the eigenvectors of G. In fact, $\{u_{r,s}^{k_x,k_y}\}$ are the eigenvectors of $-(L+U) = A - D$ because for fixed k_x, k_y we have

$$-(L+U)u_{r,s} = \frac{u_{r-1,s} + u_{r+1,s}}{(\Delta x)^2} + \frac{u_{r,s-1} + u_{r,s+1}}{(\Delta y)^2}.$$

We find that $\{u_{r,s}^{k_x,k_y}\}$ are the eigenvectors of $-(L+U) = A - D$. Since D is a diagonal matrix, the $\{u_{r,s}^{k_x,k_y}\}$ are also the eigenvectors of G.

Considering $\{u_{r,s}^{k_x,k_y}\}$ and corresponding eigenvalues (proof in next sub-problem) depend on k_x and k_y, so eigenvectors of G are independent and they can form a basis. So the errors can be expanded in terms of the Fourier modes:

$$e_{r,s}^{(n)} = \sum_{k_x,k_y} a^{(n)}(k_x, k_y)\left[\cos\left(k_x - \frac{1}{2}\right)\pi x_r\right]\left[\sin k_y \pi y_s\right]$$

2. The eigenvalues of A are $\frac{1}{(\Delta x)^2}[2cos(k_x - \frac{1}{2})\pi\Delta x - 2] + \frac{1}{(\Delta y)^2}[2cosk_y\pi\Delta y - 2]$, and $G = D^{-1}(L+U) = I - D^{-1}A$. Notice that diagonal element of A is $-\frac{2((\Delta x)^2+(\Delta y)^2)}{(\Delta x)^2(\Delta y)^2}$,

so the eigenvalues of G is:

$$\begin{aligned}\mu_J\left(k_x,k_y\right) &= 1+\frac{(\Delta x)^2(\Delta y)^2}{(\Delta x)^2+(\Delta y)^2}\{\frac{1}{(\Delta x)^2}[cos(k_x-\frac{1}{2})\pi\Delta x-1]+\frac{1}{(\Delta y)^2}[cosk_y\pi\Delta y-1]\}\\ &= \frac{(\Delta y)^2}{(\Delta x)^2+(\Delta y)^2}cos(k_x-\frac{1}{2})\pi\Delta x+\frac{(\Delta x)^2}{(\Delta x)^2+(\Delta y)^2}cosk_y\pi\Delta y\end{aligned}$$

If we choose $\Delta x=\Delta y$, we have

$$\mu_J\left(k_x,k_y\right)=\frac{1}{2}\left(\cos\left(k_x-\frac{1}{2}\right)\pi\Delta x+\cos k_y\pi\Delta y\right).$$

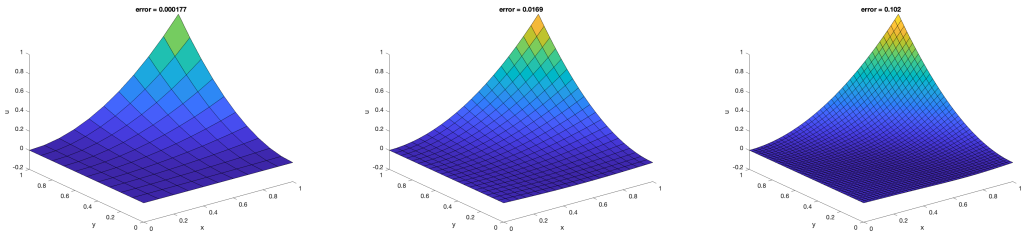
Therefore, the spectral radius of G is

$$\rho(G)=\max\{\mu_J\left(k_x,k_y\right)\}=\frac{1}{2}(cos\frac{1}{2}\pi\Delta x+cos\pi\Delta x).$$

Using Taylor expansion, we get:

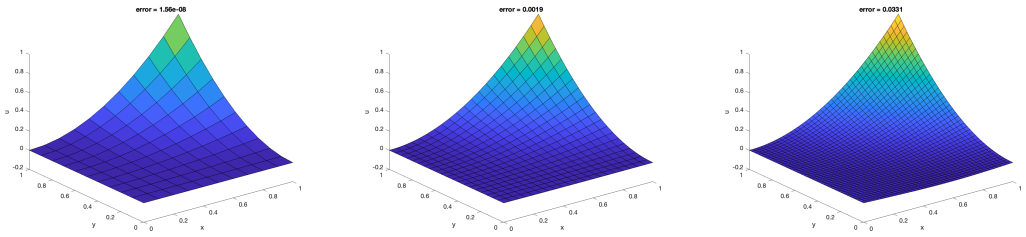
$$\rho(G)\approx 1-\frac{5}{16}(\pi\Delta x)^2$$

3. We respectively consider the number of iterations to be 200 and 500.



(a) J = 10 (b) J = 20 (c) J = 40

Figure 6: Number of iterations N = 200.



(a) J = 10 (b) J = 20 (c) J = 40

Figure 7: Number of iterations N = 500.

Appendix

```
N = 256;
t_max = 8.2;
dt = 0.005;

n_frame = ceil(t_max/dt);
x = -pi:2*pi/N:pi;
x = x';
c = 0.2 + sin(x+2).^2;
D = zeros(N+1, N+1);
for j = 1:N+1
    for l = 1:N+1
        if j == l
            D(j, l) = 0;
        else
            D(j, l) = 0.5*(-1)^(j+l)*cot((j-l)*pi/(N+1));
        end
    end
end

up = exp(-100*(x+2).^2);
upp = exp(-100*(x+2-0.2*dt).^2);


u_rec = [up, zeros(N+1, n_frame)];
for k = 1:n_frame
    u = upp - 2*dt * c .* (D*up);
    upp = up;
    up = u;

    u_rec(:, k+1) = u;
end



fig = figure;
fps = 10;
for idx = 1:fps:n_frame+1
    plot(x, u_rec(:, idx), 'LineWidth', 2)
    axis([-pi, pi, -0.1, 1.1]), xlabel('x'), ylabel('u')
    grid on
    title(['\fontsize{20} t = ', num2str((idx-1)*dt, '%.2f')])
    drawnow
    frame = getframe(fig);
    im{idx} = frame2im(frame);
end
close;


filename = 'test.gif';
for idx = 1:fps:n_frame+1
    [A, map] = rgb2ind(im{idx}, 256);
    if idx == 1
```

```
        imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',fps*dt);
    else
        imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',fps*dt);
    end
end
```

Algorithm 1: Prob_3

```
J = 40;

dx = 1/J;
dy = 1/J;

[X, Y] = meshgrid(dx*(0:J), dy*(0:J));
Z_exact = X.^2 .* Y.^2;
% surface(X, Y, Z_exact)


B = (-2*eye(J) + diag([2, ones(1, J-2)], 1) + diag(ones(1, J-1), -1))/(dx)^2;
C = (-2*eye(J-1) + diag(ones(1, J-2), 1) + diag(ones(1, J-2), -1))/(dy)^2;
A = kron(eye(J-1), B) + kron(C', eye(J));


F = zeros(J, J-1);
for i = 0:J-1
    for j = 1:J-1
        F(i+1, j) = 2*((i*dx)^2 + (j*dy)^2) - (i==J-1)*(j*dy)^2/(dx)^2 - (j==J-1)
    *(i*dx)^2/(dy)^2;
    end
end
[b, m_F, n_F] = vec(F);

x = jacobi(A, b);
% x = A\b;



U = ivec(x, m_F, n_F);
U = [zeros(J, 1), U, dx^2*(0:J-1)'.^2];
U = [U; dy^2*(0:J).^2];
surface(X, Y, U)
xlabel('x'), ylabel('y'), zlabel('u')
view(3)


error = max(abs(U-Z_exact), [], 'all');
title(['error = ', num2str(error, 3)])



function [v, m, n] = vec(A)
    [m, n] = size(A);
    v = zeros(m*n, 1);
    for j = 1:n
        v((j-1)*m+1:j*m) = A(:, j);
```

```
    end
end


function A = ivec(v, m, n)
    if length(v) ~= m*n
        error('error!')
    end

    A = zeros(m, n);
    for j = 1:n
        A(:, j) = v((j-1)*m+1:j*m);
    end
end

function x = jacobi(A, b, N)
    if nargin < 3
        N = 200;
    end
    D = diag(diag(A));
    L = -triu(A, 1);
    U = -tril(A, -1);

    T = @(x) D\(L+U)*x + D\b;
    x = zeros(size(b));
    for i = 1:N
        x = T(x);
    end
end
```

Algorithm 2: Prob_4