Question 1(A)

The Poisson equation

$$\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = g(x, y)$$

on $R = \{(x, y)|0 < x < a, 0 < y < a\}$ is subject to the following boundary conditions: $u(x, 0) = 20$, $u(x, 4) = 180$, $u(0, y) = 80$, and $u(4, y) = 0$. Choose $g(x, y) = 0$ and $a = 4$. Set up by hand the corresponding system of finite difference equations when $N = M = 4$. Solve in Matlab the system obtained.

Solution

We substitute second-order central difference approximations for each of the partial derivatives to get

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{(x_j,y_k)} = \frac{u_{j-1,k} - 2u_{j,k} + u_{j+1,k}}{\hbar^2} + \mathcal{O}(\hbar^2)$$
$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{(x_j,y_k)} = \frac{u_{j,k-1} - 2u_{j,k} + u_{j,k+1}}{\hbar^2} + \mathcal{O}(\hbar^2)$$

This yields the following equation

$$\frac{u_{j-1,k} - 2u_{j,k} + u_{j+1,k}}{h^2} + \mathcal{O}(h^2) + \frac{u_{j,k-1} - 2u_{j,k} + u_{j,k+1}}{h^2} + \mathcal{O}(h^2) = f_{j,k}$$

Simplifying, we have the following linear system we must solve

$$-w_{j-1,k} - w_{j+1,k} - w_{j,k-1} - w_{j,k+1} + 4w_{j,k} = -h^2 f_{j,k}$$

for $1 \le j \le 3$ and $1 \le k \le 3$ with the Dirichlet boundary conditions $w_{j,k} = g_{j,k}$ if $j = 0$, $j = N = 4$, $k = 0$, or $k = M = 4$. We also have $f(x, y) = 0$, so $f_{i,j} = 0$ for all i and j.

The five point-star approximation of the Laplacian gives $A\vec{w} = \vec{b}$ which is

$$\begin{aligned}
4w_{1,1} - w_{2,1} - w_{1,2} &= -h^2 f_{1,1} + g_{1,0} + g_{0,1} \\
-w_{1,1} + 4w_{2,1} - w_{3,1} - w_{2,2} &= -h^2 f_{2,1} + g_{2,0} \\
-w_{2,1} + 4w_{3,1} - w_{3,2} &= -h^2 f_{3,1} + g_{3,0} + g_{4,1} \\
-w_{1,1} + 4w_{1,2} - w_{2,2} - w_{1,3} &= -h^2 f_{1,2} + g_{0,2} \\
-w_{2,1} - w_{1,2} + 4w_{2,2} - w_{3,2} - w_{2,3} &= -h^2 f_{2,2} \\
-w_{3,1} - w_{2,2} + 4w_{3,2} - w_{3,3} &= -h^2 f_{3,2} + g_{4,2} \\
-w_{1,2} + 4w_{1,3} - w_{2,3} &= -h^2 f_{1,3} + g_{0,3} + g_{1,4} \\
-w_{2,2} - w_{1,3} + 4w_{2,3} - w_{3,3} &= -h^2 f_{2,3} + g_{2,4} \\
-w_{3,2} - w_{2,3} + 4w_{3,3} &= -h^2 f_{3,3} + g_{4,3} + g_{3,4}
\end{aligned}$$

Simplifying, we have

$$
\begin{aligned}
4w_{1,1} - w_{2,1} - w_{1,2} &= 100 \\
-w_{1,1} + 4w_{2,1} - w_{3,1} - w_{2,2} &= 20 \\
-w_{2,1} + 4w_{3,1} - w_{3,2} &= 20 \\
-w_{1,1} + 4w_{1,2} - w_{2,2} - w_{1,3} &= 80 \\
-w_{2,1} - w_{1,2} + 4w_{2,2} - w_{3,2} - w_{2,3} &= 0 \\
-w_{3,1} - w_{2,2} + 4w_{3,2} - w_{3,3} &= 0 \\
-w_{1,2} + 4w_{1,3} - w_{2,3} &= 260 \\
-w_{2,2} - w_{1,3} + 4w_{2,3} - w_{3,3} &= 180 \\
-w_{3,2} - w_{2,3} + 4w_{3,3} &= 180
\end{aligned}
$$

This gives the following system of linear equations

$$
\begin{bmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{bmatrix}
\begin{bmatrix}
w_{1,1} \\ w_{2,1} \\ w_{3,1} \\ w_{1,2} \\ w_{2,2} \\ w_{3,2} \\ w_{1,3} \\ w_{2,3} \\ w_{3,3}
\end{bmatrix}
=
\begin{bmatrix}
100 \\ 20 \\ 20 \\ 80 \\ 0 \\ 0 \\ 260 \\ 180 \\ 180
\end{bmatrix}
$$

Solving the linear system in Matlab, we have:

```
% Define the coefficient matrix A
A = [4 -1 0 -1 0 0 0 0 0;
    -1 4 -1 0 -1 0 0 0 0;
    0 -1 4 0 0 -1 0 0 0;
    -1 0 0 4 -1 0 -1 0 0;
    0 -1 0 -1 4 -1 0 -1 0;
    0 0 -1 0 -1 4 0 0 -1;
    0 0 0 -1 0 0 4 -1 0;
    0 0 0 0 -1 0 -1 4 -1;
    0 0 0 0 0 -1 0 -1 4];

% Define the right-hand side vector b
b = [100; 20; 20; 80; 0; 0; 260; 180; 180];

% Solve for the vector w
w = A \ b;

% Display the solution vector w
disp('Solution vector w:');
disp(w);

Solution vector w:
   55.7143
   43.2143
   27.1429
   79.6429
   70.0000
   45.3571
  112.8571
  111.7857
   84.2857
```

Question 1(B)

Solve in Matlab the Poisson equation with $g(x, y) = 2$ and $a = 1$, subject to the following boundary conditions $u(x, 0) = x^2$, $u(x, 1) = (x-1)^2$, $u(0, y) = y^2$, and $u(1, y) = (y-1)^2$.

Solution

We have $h = \frac{B-A}{N} = \frac{1}{4}$ and $f(x, y) = 2$, so $f_{i,j} = 2$ for all i and j.

$$
\begin{aligned}
4w_{1,1} - w_{2,1} - w_{1,2} &= -h^2 f_{1,1} + g_{1,0} + g_{0,1} \\
-w_{1,1} + 4w_{2,1} - w_{3,1} - w_{2,2} &= -h^2 f_{2,1} + g_{2,0} \\
-w_{2,1} + 4w_{3,1} - w_{3,2} &= -h^2 f_{3,1} + g_{3,0} + g_{4,1} \\
-w_{1,1} + 4w_{1,2} - w_{2,2} - w_{1,3} &= -h^2 f_{1,2} + g_{0,2} \\
-w_{2,1} - w_{1,2} + 4w_{2,2} - w_{3,2} - w_{2,3} &= -h^2 f_{2,2} \\
-w_{3,1} - w_{2,2} + 4w_{3,2} - w_{3,3} &= -h^2 f_{3,2} + g_{4,2} \\
-w_{1,2} + 4w_{1,3} - w_{2,3} &= -h^2 f_{1,3} + g_{0,3} + g_{1,4} \\
-w_{2,2} - w_{1,3} + 4w_{2,3} - w_{3,3} &= -h^2 f_{2,3} + g_{2,4} \\
-w_{3,2} - w_{2,3} + 4w_{3,3} &= -h^2 f_{3,3} + g_{4,3} + g_{3,4}
\end{aligned}
$$

$$
\begin{aligned}
4w_{1,1} - w_{2,1} - w_{1,2} &= -\frac{1}{8} + y^2 + x^2 \\
-w_{1,1} + 4w_{2,1} - w_{3,1} - w_{2,2} &= -\frac{1}{8} + x^2 \\
-w_{2,1} + 4w_{3,1} - w_{3,2} &= -\frac{1}{8} + x^2 + (y-1)^2 \\
-w_{1,1} + 4w_{1,2} - w_{2,2} - w_{1,3} &= -\frac{1}{8} + y^2 \\
-w_{2,1} - w_{1,2} + 4w_{2,2} - w_{3,2} - w_{2,3} &= -\frac{1}{8} \\
-w_{3,1} - w_{2,2} + 4w_{3,2} - w_{3,3} &= -\frac{1}{8} + (y-1)^2 \\
-w_{1,2} + 4w_{1,3} - w_{2,3} &= -\frac{1}{8} + y^2 + (x-1)^2 \\
-w_{2,2} - w_{1,3} + 4w_{2,3} - w_{3,3} &= -\frac{1}{8} + (x-1)^2 \\
-w_{3,2} - w_{2,3} + 4w_{3,3} &= -\frac{1}{8} + (x-1)^2 + (y-1)^2
\end{aligned}
$$

$$
\begin{aligned}
4w_{1,1} - w_{2,1} - w_{1,2} &= -\frac{1}{8} + (0.25)^2 + (0.25)^2 \\
-w_{1,1} + 4w_{2,1} - w_{3,1} - w_{2,2} &= -\frac{1}{8} + (0.5)^2 \\
-w_{2,1} + 4w_{3,1} - w_{3,2} &= -\frac{1}{8} + (0.75)^2 + (-0.75)^2 \\
-w_{1,1} + 4w_{1,2} - w_{2,2} - w_{1,3} &= -\frac{1}{8} + (0.5)^2 \\
-w_{2,1} - w_{1,2} + 4w_{2,2} - w_{3,2-w_{2,3}} &= -\frac{1}{8} \\
-w_{3,1} - w_{2,2} + 4w_{3,2} - w_{3,3} &= -\frac{1}{8} + (-0.5)^2 \\
-w_{1,2} + 4w_{1,3} - w_{2,3} &= -\frac{1}{8} + (0.75)^2 + (-0.75)^2 \\
-w_{2,2} - w_{1,3} + 4w_{2,3} - w_{3,3} &= -\frac{1}{8} + (-0.5)^2 \\
-w_{3,2} - w_{2,3} + 4w_{3,3} &= -\frac{1}{8} + (-0.25)^2 + (-0.25)^2
\end{aligned}
$$

$$
\begin{bmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{bmatrix}
\begin{bmatrix}
w_{1,1} \\ w_{2,1} \\ w_{3,1} \\ w_{1,2} \\ w_{2,2} \\ w_{3,2} \\ w_{1,3} \\ w_{2,3} \\ w_{3,3}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ \frac{1}{8} \\ 1 \\ \frac{1}{8} \\ -\frac{1}{8} \\ \frac{1}{8} \\ 1 \\ \frac{1}{8} \\ 0
\end{bmatrix}
$$

```
% Define the coefficient matrix
A = [4 -1 0 -1 0 0 0 0 0;
    -1 4 -1 0 -1 0 0 0 0;
     0 -1 4 0 0 -1 0 0 0;
    -1 0 0 4 -1 0 -1 0 0;
     0 -1 0 -1 4 -1 0 -1 0;
     0 0 -1 0 -1 4 0 0 -1;
     0 0 0 -1 0 0 4 -1 0;
     0 0 0 0 -1 0 -1 4 -1;
     0 0 0 0 0 -1 0 -1 4];

% Define the right-hand side vector
b = [0; 1/8; 1; 1/8; -1/8; 1/8; 1; 1/8; 0];

% Solve the system of linear equations
w = A\b;

% Display the solution
disp('Solution vector w:');
disp(w);

Solution vector w:
    0.0859
    0.1719
    0.3359
    0.1719
    0.1406
```

```
0.1719
0.3359
0.1719
0.0859
```

Question 2(A)

The heat equation $\frac{du}{dt} = \frac{d^2u}{x^2}$ for $R = \{(t,x)|0 < t < 1, 0 < x < \pi\}$ is subject to the initial and boundary conditions $u(0,t) = 0$, $u(\pi,t) = 0$, and $u(x,0) = sin(x)$. The exact solution of this IBVP is $u(x,t) = e^{-t}sin(x)$. Set up by hand the BTCS discretization scheme for $\delta x = \frac{\pi}{4}$ and $\delta t = \frac{1}{3}$. Using this scheme, write a table with the values of the approximate solution at $t = 1$, the exact solution, and absolute error at the grid points obtained.

Solution

```
% Define the constants
pi_value = pi;

% Num Time Steps (delta_t)
delta_t = 1/3;

b = pi_value;
a = 0;

D = 1;

exact_solution = @(x) exp(-1) * sin(x);

delta_x = (b-a) / 4;

% Num Space Steps (delta_x = (B-A)/N)
M = (b-a) / delta_x;

lambda = D * delta_t / (delta_x^2);

% Set up evolution matrix
A = diag(1 + 2*lambda*ones(M-1,1) ) - diag(lambda*ones(M-2,1),1);
A = A - diag(lambda*ones(M-2,1),-1);

% Compute starting x_j values (there are M many of them)
x = zeros(1, M-1);
b_vector = zeros(1, M-1);
for k = 1:M-1
    x(k) = delta_x * k;
    b_vector(k) = sin(x(k));

end
b_vector = b_vector.';

% Solve N linear systems until final one @ t=1
N = 1 / delta_t;

for j = 1:N
    approx_x = A \ b_vector;
    b_vector = approx_x;
    disp(j);
    disp(b_vector);
end
exact_soln = exact_solution(x).';
```

```
     1

    0.5371
```

```
0.7596
0.5371

 2

0.4080
0.5769
0.4080

 3

0.3099
0.4382
0.3099
```

x_j	Approximate Value	Exact Solution	Abs. Error
0.7854	0.3099	0.260	0.0499
1.5708	0.4382	0.368	0.0702
2.3562	0.3099	0.260	0.0499

Question 2(B)

Write Matlab to numerically verify that the FTCS method is second-order accurate in space.

Solution

```
% Define the constants
pi_value = pi;

% Num Time Steps (delta_t)
delta_t = 0.00005;

b = pi_value;
a = 0;

D = 1;

exact_solution = @(x) exp(-1) * sin(x);

num_stepsizes = 5;
max_abs_error = zeros(1, num_stepsizes);

for i = 1:num_stepsizes

    delta_x = (b-a) / (4 * 2^(i - 1));

    % Num Space Steps (delta_x = (B-A)/N)
    M = (b-a) / delta_x;

    lambda = D * delta_t / (delta_x^2);

    % Set up evolution matrix
    A = diag(1 + 2*lambda*ones(M-1,1) ) - diag(lambda*ones(M-2,1),1);
    A = A - diag(lambda*ones(M-2,1),-1);

    % Compute starting x_j values (there are M many of them)
    x = zeros(1, M-1);
    b_vector = zeros(1, M-1);
    for k = 1:M-1
        x(k) = delta_x * k;
        b_vector(k) = sin(x(k));

    end
    b_vector = b_vector.';

    % Solve N linear systems until final one @ t=1
    N = 1 / delta_t;

    for j = 1:N
        approx_x = A \ b_vector;
        b_vector = approx_x;
    end
    exact_soln = exact_solution(x).';

    % Store x_j value with max absolute error
    max_abs_error(i) = max(abs(approx_x - exact_soln));
```

```
disp('max_abs_error: ');
disp(max_abs_error);

end

max_abs_error:
    0.0190    0.0047    0.0012    0.0003    0.0001
```

N	Max Abs Error	Error Ratio from Last Column
4	0.0190	-
8	0.0047	4.04
16	0.0012	3.92
32	0.0003	4.00
64	0.0001	3.72

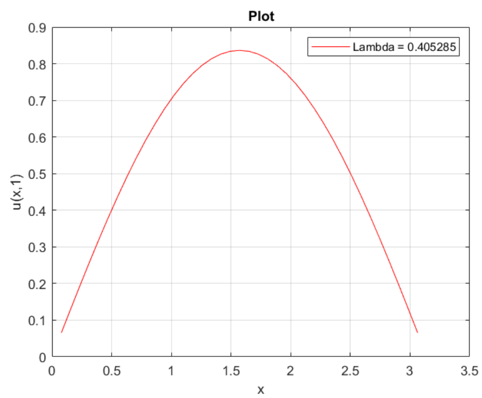
Question 3

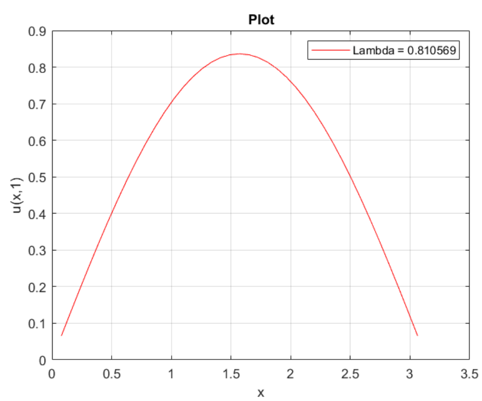
Consider the following parabolic equation $\frac{du}{dt} = \frac{d^2u}{dx^2} - (x+t^2)e^{-xt}$ subject to $u(0,t) = 1$, $u(\pi,t) = e^{-\pi t}$, $u(x,0) = 1 + \sin(x)$. Verify numerically that the Crank-Nicholson method is unconditionally stable.

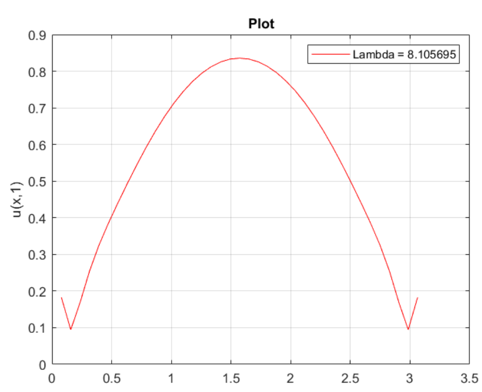
Solution

```
% Define the constants
pi_value = pi;

% Num Time Steps (delta_t)
delta_t = [1/100 1/50 1/10];

delta_x = 1/40;

b = 1;
a = 0;

D = 1;
i = 3;

% Num Space Steps (delta_x = (B-A)/N)
M = (b-a) / delta_x;
lambda = D * delta_t(i) / (delta_x^2);

% Create tridiagonal matrix A
A = diag(2*ones(M-1,1)) + diag(-1*ones(M-2,1), -1) + diag(-1*ones(M-2,1), 1);

% Create identity matrix
I = eye(M-1);

% Calculate (I + lambda * A)^-1
inv_term = inv(I + lambda * A);

% Calculate -I + 2 * (I + lambda * A)^-1
result = -I + 2 * inv_term;


% Compute starting x_j values (there are M many of them)
x = zeros(1, M-1);
b_vector = zeros(1, M-1);
for k = 1:M-1
    x(k) = delta_x * k;
    b_vector(k) =  1 + sin(x(k));

end
b_vector = b_vector.';

% Solve N linear systems until final one @ t=1
N = 1 / delta_t(i);
for j = 1:N
    approx_x = A \ b_vector;
    b_vector = approx_x;
    %disp(j);
    %disp('b_vector: ');
    %disp(b_vector);
end
```

```
%disp('x: ');
%disp(x);
%disp('b_vector: ');
%disp(b_vector);

% Plotting
plot(x.', b_vector, 'r');
xlabel('x');
ylabel('u(x,1)');
title('Plot');
legend(sprintf('Lambda = %f', lambda));
grid on;
```







Question 4

Solution

We have $a(x,t,u) = \frac{1+x^2}{(1+x^2)^2+2tx}$ for all t_n, which implies that the characteristic of the PDE, $x(t)$, travels from left to right for all t_n. So, the upwind difference method will always use a backward difference for the space derivative.

We also have $g(x,t,u) = 0$. We must take A and B such that $\frac{1}{2} < x < 1$ is included in the interval $A \leq x \leq B$, and so that we can integrate until $t = 2$, as desired. The exact solution, as given, is $u(x,t) = u_0(x - \frac{t}{1+x^2})$.

```
% Dr. Ilie's advectionFD.m code modified

function advectionFD(A,B,dx,dt,tmax)

% define num space steps
N = (B-A)/dx;

% define num time steps
M = tmax/dt;

% define space grid points
x = linspace(A,B,N+1);

% define  time grid points from t=0 to t=3
t = dt*(0:M);

% define initial condition
w(1:N+1, 1) = 0;
w(x < 1 & x > 1/2) = 1;

% define a(x,t,u)= 2*t
% a is an N by M matrix
for j=1:N
    for k=1:M
        a(j,k) = (1+x(j).^2) / ( (1+x(j).^2).^2 + 2.*t(k).*x(j) );
    end
end

% define the right hand side function g=0
g(1:N+1,1:M+1) = 0;

for n=1:M      % advance in time from n = 1 (initial condition) to n = M
    for j=2:N  % advance in space
        if (a(j, n) > 0)  % changed from n to n+1
            w(j, n+1) = (1 - a(j, n)*dt/dx ) * w(j, n) + a(j, n)*dt/dx*w(j - 1, n) + g(j, n)*dt;
        else
            disp('test');
            disp(a(j, n));
            disp(j);
            disp(n);
            w(j, n+1) = (1 + a(j, n)*dt/dx) * w(j, n) - a(j, n)*dt/dx*w(j + 1, n) + g(j, n)*dt;
        end
    end
end

%  plot the solution at t=1, t=2, t=3
xlabel('x');
ylabel('u(x,t)');
```

```
title('Solution of advection equation at t=1 (b) and t=2 (r)');
plot(x, w(1:N+1,t==1), 'b', ...
    x, w(1:N+1,t==2),'r');
grid on;
```