

Advanced Numerical Methods for PDEs

Homework 3

Francisco Castillo

April 30, 2021

Consider the 2D problem

$$\begin{aligned}\partial_t u(x, y, t) + \partial_x f^x + \partial_y f^y &= 0, \quad x, y \in [0, 1] \\ f^x &= u - \partial_x u, \quad f^y = -\partial_y u\end{aligned}$$

with boundary conditions

$$\begin{aligned}f^x(x = 0, y, t) &= 0, \quad f^x(x = 1, y, t) = 0, \\ f^y(x, y = 0, t) &= 0, \quad f^y(x, y = 1, t) = 0,\end{aligned}$$

and initial conditions

$$u(x, y, t = 0) = u^I(x, y) = \delta(x - 0.5)\delta(y - 0.5)$$

Problem 1

1. Write a code which solves the problem using the implicit Cranck - Nicholson scheme (the trapezoidal rule in time), using an $M \times N$ grid with $\Delta x = \Delta y$. Explain how you solve the linear systems, using sparse Gauss elimination.

Solution: We can represent the PDE as

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0$$

Next, we discretize the equation using Crank-Nicholson in time, central differences

in space:

$$\begin{aligned} \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + \frac{1}{2} \left(\frac{u_{i+1,j}^{n+1} - u_{i-1,j}^{n+1}}{2\Delta x} + \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} \right) \\ - \frac{1}{2} \left(\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} \right) \\ - \frac{1}{2} \left(\frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) = 0 \end{aligned}$$

Regrouping the $n + 1$ terms on the left and the n terms on the right we obtained

$$\begin{aligned} u_{i,j}^{n+1} + \frac{C}{4} (u_{i+1,j}^{n+1} - u_{i-1,j}^{n+1}) - \frac{\mu_x}{2} (u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}) - \frac{\mu_y}{2} (u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}) \\ = u_{i,j}^n - \frac{C}{4} (u_{i+1,j}^n - u_{i-1,j}^n) + \frac{\mu_x}{2} (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{\mu_y}{2} (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n), \end{aligned}$$

with $C = \Delta t / \Delta x$, $\mu_x = \Delta t / \Delta x^2$, and $\mu_y = \Delta t / \Delta y^2$.

Before we continue, let us define the grid with M and N horizontal and vertical cells, respectively. Note that the domain is then defined by $(M + 1)(N + 1)$ grid nodes. Hence the coordinates $x_i = (i - 2)\Delta_x$ with $i \in [1, M + 3]$ and $y_j = (j - 2)\Delta_y$ with $j \in [1, N + 3]$ include the *ghost nodes* at indices 1 and $M + 3$, $N + 3$ (chosen this way to match the *MATLAB* array indexing). We can collapse this 2D array into 1 column vector by stacking the columns one after another. We will obtain a vector of dimensions $(M + 3)(N + 3)$, including ghost nodes. Thus, our solution vector is $U_k = U_{i+(M+3)(j-1)} = u_{i,j}$. It is easy to show that

$$\begin{aligned} u_{i\pm 1,j} &= U_{i\pm 1+(M+3)(j-1)} = U_{k\pm 1}, \\ u_{i,j\pm 1} &= U_{i+(M+3)(j\pm 1-1)} = U_{k\pm(M+3)}. \end{aligned}$$

Then, we can turn our discretization to be in terms of the solution vector U ,

$$\begin{aligned} U_k^{n+1} + \frac{C}{4} (U_{k+1}^{n+1} - U_{k-1}^{n+1}) - \frac{\mu_x}{2} (U_{k+1}^{n+1} - 2U_k^{n+1} + U_{k-1}^{n+1}) - \frac{\mu_y}{2} (U_{k+M+3}^{n+1} - 2U_k^{n+1} + U_{k-M-3}^{n+1}) \\ = U_k^n - \frac{C}{4} (U_{k+1}^n - U_{k-1}^n) + \frac{\mu_x}{2} (U_{k+1}^n - 2U_k^n + U_{k-1}^n) + \frac{\mu_y}{2} (U_{k+M+3}^n - 2U_k^n + U_{k-M-3}^n). \end{aligned}$$

Reorganizing the equation we get

$$\begin{aligned} -\frac{\mu_y}{2} U_{k-M-3}^{n+1} - \left(\frac{C}{4} + \frac{\mu_x}{2} \right) U_{k-1}^{n+1} + (1 + \mu_x + \mu_y) U_k^{n+1} + \left(\frac{C}{4} - \frac{\mu_x}{2} \right) U_{k+1}^{n+1} - \frac{\mu_y}{2} U_{k+M+3}^{n+1} \\ = \frac{\mu_y}{2} U_{k-M-3}^n + \left(\frac{C}{4} + \frac{\mu_x}{2} \right) U_{k-1}^{n+1} + (1 - \mu_x - \mu_y) U_k^{n+1} + \left(\frac{\mu_x}{2} - \frac{C}{4} \right) U_{k+1}^{n+1} + \frac{\mu_y}{2} U_{k+M+3}^{n+1}, \end{aligned}$$

$$\begin{aligned} a_{-M-3} U_{k-M-3}^{n+1} + a_{-1} U_{k-1}^{n+1} + a_0 U_k^{n+1} + a_{+1} U_{k+1}^{n+1} + a_{M+3} U_{k+M+3}^{n+1} \\ = b_{-M-3} U_{k-M-3}^n + b_{-1} U_{k-1}^{n+1} + b_0 U_k^{n+1} + b_{+1} U_{k+1}^{n+1} + b_{M+3} U_{k+M+3}^{n+1}, \end{aligned}$$

This system of equations can be represented in matrix form, $AU^{n+1} = BU^n$, where A and B are sparse matrices defined by the 5 a diagonals and 5 b diagonals given in the equation above. To solve the system we use the *MATLAB* command backslash operator $U^{n+1} = A \backslash BU^n$. Lastly, we need to implement the boundary conditions after updating the solution. The boundary conditions in index form are:

- Bottom boundary: $\partial_y u|_{y=0} = 0$:

$$\begin{aligned} \left. \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \right|_{j=2} &= 0 \\ \frac{u_{i,3} - u_{i,1}}{2\Delta y} &= 0 \\ u_{i,1} = u_{i,3} &\Rightarrow \boxed{U_i = U_{i+2(M+3)}} \end{aligned}$$

- Top boundary: $\partial_y u|_{y=1} = 0$:

$$\begin{aligned} \left. \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \right|_{j=N+2} &= 0 \\ \frac{u_{i,N+3} - u_{i,N+1}}{2\Delta y} &= 0 \\ u_{i,N+3} = u_{i,N+1} &\Rightarrow \boxed{U_{i+(N+2)(M+3)} = U_{i+N(M+3)}} \end{aligned}$$

- Left boundary: $u|_{x=0} - \partial_x u|_{x=0} = 0$:

$$\begin{aligned} u_{i,j}|_{i=2} - \left. \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \right|_{i=2} &= 0 \\ u_{2,j} - \frac{u_{3,j} - u_{1,j}}{2\Delta x} &= 0 \\ u_{1,j} &= u_{3,j} - 2\Delta x u_{2,j} \\ \boxed{U_{1+(M+3)(j-1)} = U_{3+(M+3)(j-1)} - 2\Delta x U_{2+(M+3)(j-1)}} \end{aligned}$$

- Right boundary: $u|_{x=1} - \partial_x u|_{x=1} = 0$:

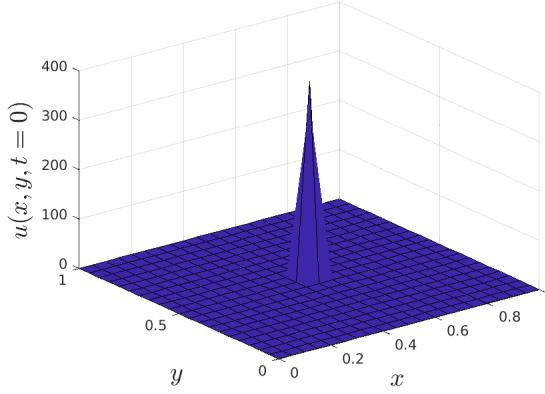
$$\begin{aligned} u_{i,j}|_{i=M+2} - \left. \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \right|_{i=M+2} &= 0 \\ u_{M+2,j} - \frac{u_{M+3,j} - u_{M+1,j}}{2\Delta x} &= 0 \\ u_{M+3,j} &= u_{M+1,j} + 2\Delta x u_{M+2,j} \\ \boxed{U_{(M+3)j} = U_{M+1+(M+3)(j-1)} + 2\Delta x U_{M+2+(M+3)(j-1)}} \end{aligned}$$

The four boundary conditions are imposed after updating the solution by solving $U^{n+1} = A \backslash BU^n$ every time-step.

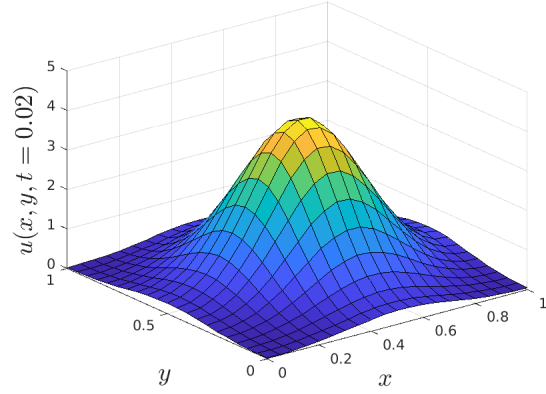
Problem 2

1. Using the Crank - Nicholson scheme with a time and spatial step $\Delta t = \Delta x = 0.01$ for $0 < t < 1$. Solve the linear equations resulting from the implicit discretization in *MATLAB* sparse mode.

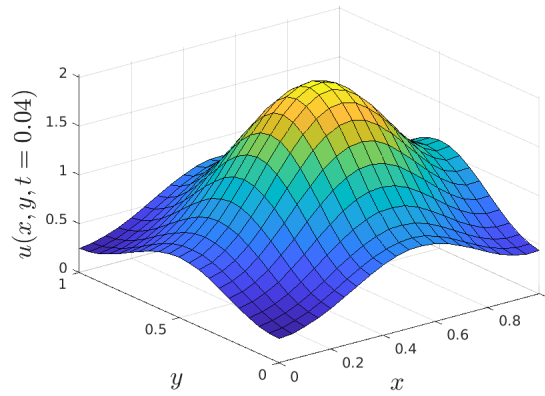
Solution:



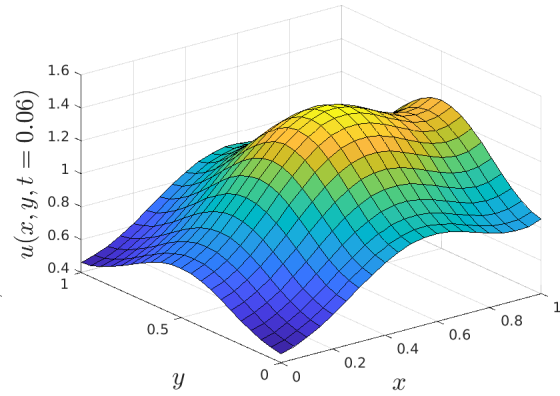
(a) $t = 0$.



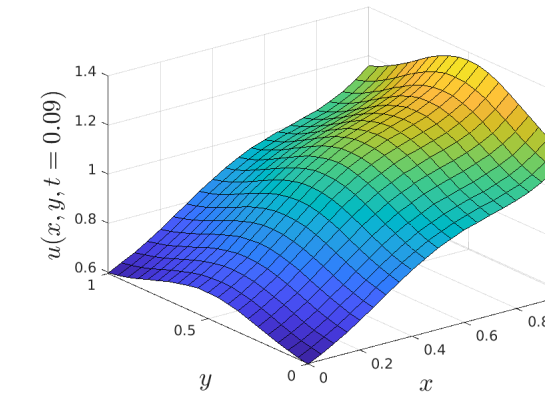
(b) $t = 0.02$.



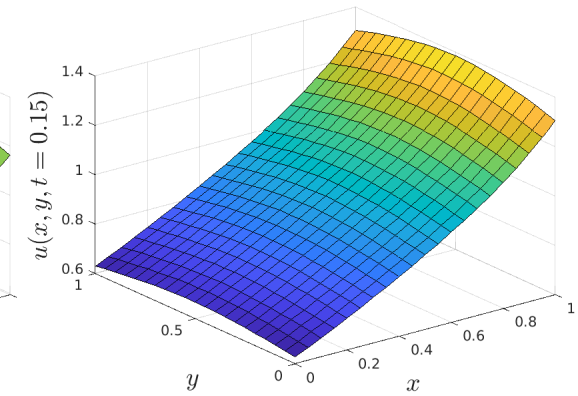
(c) $t = 0.04$.



(d) $t = 0.06$.



(e) $t = 0.09$.



(f) $t = 0.15$.

Figure 1: Solution $u(x, t)$ against x and t for different number of realizations.

```
1 clear all, close all, clc
2
```

```

3 format short
4
5 % Number of intervals
6 M = 20; % In x-direction
7 N = 20; % In Y-direction
8
9 % Define limits of domain
10 x_0 = 0;
11 x_F = 1;
12 y_0 = 0;
13 y_F = 1;
14 T = 0.2;
15
16 % Step sizes
17 dx = (x_F - x_0) / M;
18 dy = (y_F - y_0) / N;
19 dt = 0.0001;
20
21 % Define grid (including ghost nodes)
22 x = linspace (x_0-dx, x_F+dx, M+3); % x(1) and x(M+3) are ghost
    nodes
23 y = linspace (y_0-dy, y_F+dy, N+3); % y(1) and y(N+3) are ghost
    nodes
24
25 % Define useful constants
26 C = dt/dx;
27 mu_x = dt/(dx^2);
28 mu_y = dt/(dy^2);
29
30 % Define initial condition
31 U0 = kron((round(x,8) == 0.5),((round(y,8) == 0.5)))/dx/dy;
32 plot_sol(U0,x,y,0)
33
34 % Define big matrices size, including ghost nodes
35 S = (M + 3)*(N + 3);
36
37 %% Construct Matrix A
38 % Diagonal component
39 d = diag((1 + mu_x + mu_y) * ones(S,1));
40 % 1-lower diagonal component
41 l1 = diag((-C/4 - mu_x/2) * ones(S-1,1), -1);
42 % 1-upper diagonal component
43 u1 = diag(( C/4 - mu_x/2) * ones(S-1,1), 1);
44 % (M+3)-lower diagonal component
45 lM3 = diag(-mu_y/2 * ones(S-M-3,1), -M-3);
46 % (M+3)-upper diagonal component
47 uM3 = diag(-mu_y/2 * ones(S-M-3,1), M+3);
48 % Add them together
49 A = sparse(d + u1 + l1 + lM3 + uM3);
50
51 %% Construct Matrix B
52 % Diagonal component

```

```

53 d = diag((1 - mu_x - mu_y) * ones(S,1));
54 % 1-lower diagonal component
55 l1 = diag(( C/4 + mu_x/2) * ones(S-1,1), -1);
56 % 1-upper diagonal component
57 u1 = diag((-C/4 + mu_x/2) * ones(S-1,1), 1);
58 % (M+3)-lower diagonal component
59 lM3 = diag(mu_y/2 * ones(S-M-3,1), -M-3);
60 % (M+3)-upper diagonal component
61 uM3 = diag(mu_y/2 * ones(S-M-3,1), M+3);
62 % Add them together
63 B = sparse(d + u1 + l1 + lM3 + uM3);
64
65 %% Compute solution
66
67 t_plots = 0.01:0.01:T
68
69 t = 0;
70 tstep = 0;
71 plot_idx = 1;
72 % Initialize
73 U = U0;
74
75 i = 1:M+3; % Horizontal array index
76 j = 1:N+3; % Vertical array index
77 while t < T
78
79     % Advance solution
80     rhs = B*U;
81     U = A\rhs;
82
83     % Impose boundary conditions
84     U(i) = U(i + 2*(M+3)); % Bottom
85     U(i + (N+2)*(M+3)) = U(i + N*(M+3)); % Top
86     U(1 + (M+3)*(j-1)) = U(3 + (M+3)*(j-1)) - 2*dx*U(2 + (M+3)*(j-1)
); % Left
87     U(M+3 + (M+3)*(j-1)) = U(M+1 + (M+3)*(j-1)) + 2*dx*U(M+2 + (M+3)
*(j-1)); % Right
88
89     % Advance time
90     t = round(t + dt,4); % avoid floating point error to compare
with ismember
91     tstep = tstep + 1;
92
93     if (mod(tstep,100) == 0)
94         plot_sol(U,x,y,t)
95         pause(0.15)
96     end
97
98 end
99
100 function plot_sol(U,x,y,t)
101     labelfontsize = 18;

```

```

102     Lx = length(x);
103     Ly = length(y);
104     u = reshape(U, [Lx Ly]);
105     figure(1)
106
107     surf(x(2:Ly-1),y(2:Lx-1),u(2:Lx-1,2:Ly-1)')
108
109     xlabel('$x$', 'interpreter', 'latex', 'fontsize', labelfontsize)
110     ylabel('$y$', 'interpreter', 'latex', 'fontsize', labelfontsize)
111     zlabel(append('$u(x, y, t = ', num2str(t), ')$'), 'interpreter', '
112     latex', 'fontsize', labelfontsize)
113     figName = create_figName(t);
114     saveas(gcf, figName, 'png')
115 end
116
117 function figName = create_figName(t)
118     if t == 0
119         exponent = 0;
120         base = 00;
121     else
122         exponent = -2;
123         base = t*100;
124     end
125
126     path = '../figures/';
127
128     if base > 9
129         figName = append(path, 'p2_u_t', num2str(base), 'e', num2str(
130         exponent));
131     else
132         figName = append(path, 'p2_u_t0', num2str(base), 'e', num2str(
133         exponent));
134     end
135 end

```


Consider the Schroedinger equation

$$i\hbar \frac{\partial \psi(x, t)}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi(x, t) + V(x) \psi(x, t).$$

Here \hbar is Plancks constant, m is the mass of the particle, and $V(x)$ is the external potential energy. ψ is the wave function. The particle density is given by $u(x, t) = |\psi(x, t)|^2$. Assume units for space, time and mass such that $\hbar = m = 1$ holds.

Problem 3

1. In these units derive the equations for a spectral Galerkin method on the interval $x \in [\pi, p\pi]$ with periodic boundary conditions, using the basis functions

$$\phi_n(x) = e^{inx}, \quad n = -N : N,$$

and the weighted complex $L2$ scalar product $\langle u, v \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} u^* v dx$ for a potential energy given by

$$V(x) = \begin{cases} 0 & -\pi < x < 0 \\ 1 & 0 < x < \pi \end{cases}$$

Solution: Multiplying Schroedinger's equation by the complex constant i and making $\hbar = m = 1$ we obtain

$$\frac{\partial \psi(x, t)}{\partial t} = \frac{i}{2} \frac{\partial^2}{\partial x^2} \psi(x, t) - iV(x) \psi(x, t).$$

Let us approximate the solution using the basis functions $\phi_n(x)$ given,

$$\psi(x, t) \approx \psi_N(x, t) = \sum_{n=-N}^N c_n(t) \phi_n(x) = \sum_{n=-N}^N c_n(t) e^{inx}$$

Define

$$\begin{aligned} F(\psi_N) &:= \frac{i}{2} \frac{\partial^2}{\partial x^2} \psi_N(x, t) - iV(x) \psi_N(x, t) \\ &= \frac{i}{2} \frac{\partial^2}{\partial x^2} \sum_{n=-N}^N c_n(t) e^{inx} - iV(x) \sum_{n=-N}^N c_n(t) e^{inx} \\ &= -\frac{i}{2} \sum_{n=-N}^N c_n(t) n^2 e^{inx} - iV(x) \sum_{n=-N}^N c_n(t) e^{inx} \\ &= -i \sum_{n=-N}^N \left[c_n(t) \left(V(x) + \frac{n^2}{2} \right) e^{inx} \right]. \end{aligned}$$

Further,

$$\begin{aligned}
b_m &= \langle \phi_m, F(\psi_N) \rangle \\
&= -i \frac{1}{2\pi} \int_{-\pi}^{\pi} \left\{ e^{-imx} \sum_{n=-N}^N \left[c_n(t) \left(V(x) + \frac{n^2}{2} \right) e^{inx} \right] \right\} dx \\
&= -i \sum_{n=-N}^N c_n(t) \int_{-\pi}^{\pi} \frac{1}{2\pi} \left(V(x) + \frac{n^2}{2} \right) e^{i(n-m)x} dx \\
&= -i \sum_{n=-N}^N c_n(t) \left[\int_{-\pi}^{\pi} \frac{1}{2\pi} V(x) e^{i(n-m)x} dx + \int_{-\pi}^{\pi} \frac{1}{2\pi} \frac{n^2}{2} e^{i(n-m)x} dx \right] \\
&= -i \sum_{n=-N}^N c_n(t) \left[I_{m,n}^V + \frac{n^2}{2} \delta_{m,n} \right],
\end{aligned}$$

where $I_{m,n}^V = \frac{1}{2\pi} \int_{-\pi}^{\pi} V(x) e^{i(n-m)x} dx$ and $\delta_{m,n}$ is the Kronecker delta. Both $I_{m,n}^V$, and $\delta_{m,n}$ can be expressed in matrix form. Hence,

$$\vec{b}(t) = A \vec{c}(t),$$

where $A = -i \left[I_{m,n}^V + \frac{n^2}{2} \delta_{m,n} \right]$. Since $\vec{c}'(t) = \vec{b}(t)$, we conclude

$$\vec{c}'(t) = A \vec{c}(t),$$

where the entries of A are determined by the entries of $I_{m,n}^V$, and $\delta_{m,n}$, which are detailed below:

$$\begin{aligned}
I_{m,n}^V &= \frac{1}{2\pi} \int_{-\pi}^{\pi} V(x) e^{i(n-m)x} dx \\
&= \frac{1}{2\pi} \int_0^{\pi} e^{i(n-m)x} dx,
\end{aligned}$$

since $V(x) = 0$ from $-\pi$ to 0 . Hence,

- For $n = m$:

$$\begin{aligned}
I_{m,n}^V &= \frac{1}{2\pi} \int_0^{\pi} e^{i(n-m)x} dx \\
&= \frac{1}{2\pi} \int_0^{\pi} dx \\
&= \frac{1}{2}.
\end{aligned}$$

- For $n \neq m$:

$$\begin{aligned}
 I_{m,n}^V &= \frac{1}{2\pi} \int_0^\pi e^{i(n-m)x} dx \\
 &= \frac{1}{2\pi} \frac{i(1 - e^{i(n-m)\pi})}{(n-m)} \\
 &= \frac{i}{2\pi(n-m)} [1 - (-1)^{(n-m)}] .
 \end{aligned}$$

Problem 4

1. Solve the SE for $0 < t < 1$ for $N = 3, 10, 50$ with a time step Δt such that

$$\Delta t \|A\|^2 = \Delta t \sqrt{(A^H A)} = 1$$

holds. Here $\rho(A^H A)$ is the spectral radius $\rho(A^H A) = \max_n |\lambda_n|$ with λ_n the eigenvalues of $A^H A$ and A^H the hermitian of the complex matrix A : $A^H = (A^T)^*$. Use the MATLAB function `eig` to compute the eigenvalues. Use

$$\psi^I(x) = \begin{cases} 1 & -\pi < x < -\frac{\pi}{2} \\ 0 & -\frac{\pi}{2} < x < \pi \end{cases}$$

as an initial condition, and the Crank - Nicholson scheme for the time discretization.

Solution: From the previous problem, recover the equation

$$\vec{c}'(t) = A \vec{c}(t),$$

and apply the Crank-Nicholson scheme,

$$\frac{\vec{c}^{k+1} - \vec{c}^k}{\Delta t} = A \frac{1}{2} (\vec{c}^{k+1} + \vec{c}^k),$$

which yields

$$\vec{c}^{k+1} - A \frac{\Delta t}{2} \vec{c}^{k+1} = \vec{c}^k + A \frac{\Delta t}{2} \vec{c}^k.$$

Solving for \vec{c}^{k+1} ,

$$\vec{c}^{k+1} = \left(\mathbb{I} - A \frac{\Delta t}{2} \right)^{-1} \left(\mathbb{I} + A \frac{\Delta t}{2} \right) \vec{c}^k.$$