Advanced Numerical Methods for PDEs

Homework 3

Francisco Castillo

May 1, 2021

Consider the 2D problem

$$\partial_t u(x,y,t) + \partial_x f^x + \partial_y f^y = 0, \quad x, y \in [0,1]$$
$$f^x = u - \partial_x u, \;\; f^y = -\partial_y u$$

with boundary conditions

$$f^x(x=0,y,t) = 0, \quad f^x(x=1,y,t) = 0,$$
$$f^y(x,y=0,t) = 0, \quad f^y(x,y=1,t) = 0,$$

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:

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

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

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

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 indeces 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

$$u_{i\pm1,j}=U_{i\pm1+(M+3)(j-1)}=U_{k\pm1},$$
$$u_{i,j\pm1}=U_{i+(M+3)(j\pm1-1)}=U_{k\pm(M+3)}.$$

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

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

Reorganizing the equation we get

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

$$a_{-M-3}U_{k-M-3}^{n+1}+a_{-1}U_{k-1}^{n+1}+a_0U_k^{n+1}+a_{+1}U_{k+1}^{n+1}+a_{M+3}U_{k+M+3}^{n+1}$$
$$=b_{-M-3}U_{k-M-3}^{n}+b_{-1}U_{k-1}^{n+1}+b_0U_k^{n+1}+b_{+1}U_{k+1}^{n+1}+b_{M+3}U_{k+M+3}^{n+1},$$

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: $\partial_y u\big|_{y=0} = 0$:

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

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

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

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

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

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

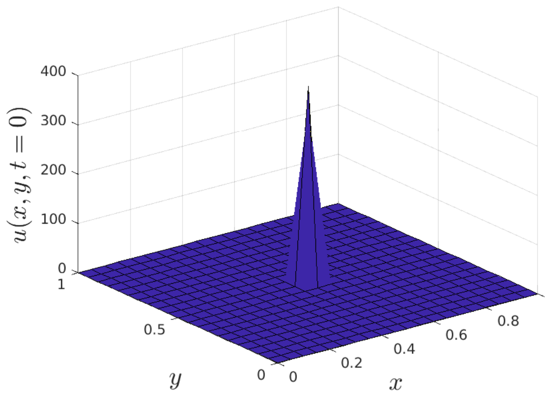
$$\begin{aligned} u_{i,j}\big|_{i=M+2} - \left.\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\right|_{i=M+2} &= 0 \\ u_{M+2,j} - \frac{u_{M+3,j} - u_{M+1,j}}{2\Delta x} &= 0 \\ u_{M+3,j} &= u_{M+1,j} + 2\Delta x u_{M+2,j} \\ &\boxed{U_{(M+3)j} = U_{M+1+(M+3)(j-1)} + 2\Delta x U_{M+2+(M+3)(j-1)}} \end{aligned}$$

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

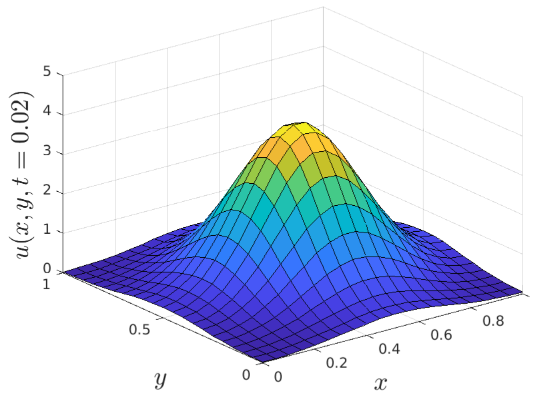
Problem 2

1. Using the Cranck - 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.

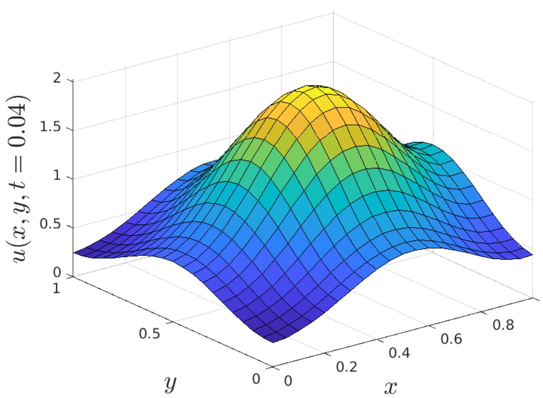
Solution:



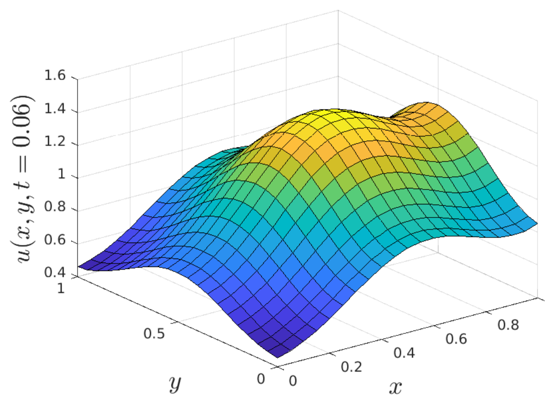
(a) $t = 0$.



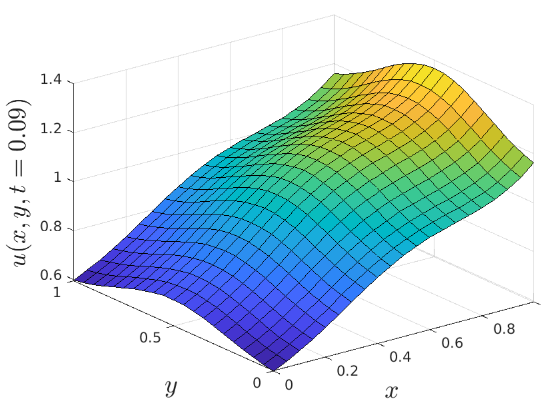
(b) $t = 0.02$.



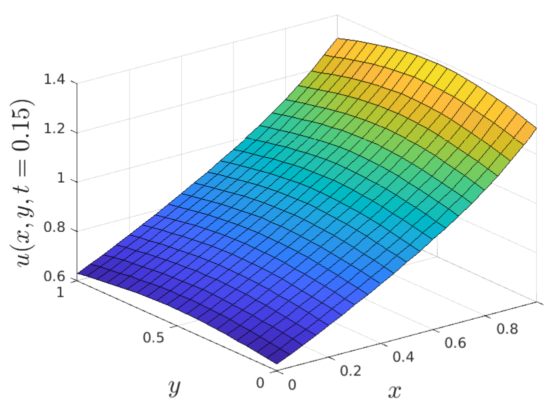
(c) $t = 0.04$.



(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.

```
clear all, close all, clc

```

```
format short

% Number of intervals
M = 20; % In x-direction
N = 20; % In Y-direction

% Define limits of domain
x_0 = 0;
x_F = 1;
y_0 = 0;
y_F = 1;
T = 0.2;

% Step sizes
dx = (x_F - x_0) / M;
dy = (y_F - y_0) / N;
dt = 0.0001;

% Define grid (including ghost nodes)
x = linspace (x_0-dx, x_F+dx, M+3); % x(1) and x(M+3) are ghost nodes
y = linspace (y_0-dy, y_F+dy, N+3); % y(1) and y(N+3) are ghost nodes

% Define useful constants
C = dt/dx;
mu_x = dt/(dx^2);
mu_y = dt/(dy^2);

% Define initial condition
U0 = kron((round(x,8) == 0.5),((round(y,8) == 0.5)))'/dx/dy;
plot_sol(U0,x,y,0)

% Define big matrices size, including ghost nodes
S = (M + 3)*(N + 3);

%% Construct Matrix A
% Diagonal component
d = diag((1 + mu_x + mu_y) * ones(S,1));
% 1-lower diagonal component
l1 = diag((-C/4 - mu_x/2) * ones(S-1,1), -1);
% 1-upper diagonal component
u1 = diag(( C/4 - mu_x/2) * ones(S-1,1), 1);
% (M+3)-lower diagonal component
lM3 = diag(-mu_y/2 * ones(S-M-3,1), -M-3);
% (M+3)-upper diagonal component
uM3 = diag(-mu_y/2 * ones(S-M-3,1), M+3);
% Add them together
A = sparse(d + u1 + l1 + lM3 + uM3);

%% Construct Matrix B
% Diagonal component
```

```
d = diag((1 - mu_x - mu_y) * ones(S,1));
% 1-lower diagonal component
l1 = diag(( C/4 + mu_x/2) * ones(S-1,1), -1);
% 1-upper diagonal component
u1 = diag((-C/4 + mu_x/2) * ones(S-1,1), 1);
% (M+3)-lower diagonal component
lM3 = diag(mu_y/2 * ones(S-M-3,1), -M-3);
% (M+3)-upper diagonal component
uM3 = diag(mu_y/2 * ones(S-M-3,1), M+3);
% Add them together
B = sparse(d + u1 + l1 + lM3 + uM3);

%% Compute solution

t_plots = 0.01:0.01:T

t = 0;
tstep = 0;
plot_idx = 1;
% Initialize
U = U0;

i = 1:M+3; % Horizontal array index
j = 1:N+3; % Vertical array index
while t < T

    % Advance solution
    rhs = B*U;
    U = A\rhs;

    % Impose boundary conditions
    U(i) = U(i + 2*(M+3)); % Bottom
    U(i + (N+2)*(M+3)) = U(i + N*(M+3)); % Top
    U(1 + (M+3)*(j-1)) = U(3 + (M+3)*(j-1)) - 2*dx*U(2 + (M+3)*(j-1)
    ); % Left
    U(M+3 + (M+3)*(j-1)) = U(M+1 + (M+3)*(j-1)) + 2*dx*U(M+2 + (M+3)
    *(j-1)); % Right

    % Advance time
    t = round(t + dt,4); % avoid floating point error to compare
    with ismember
    tstep = tstep + 1;

    if (mod(tstep,100) == 0)
        plot_sol(U,x,y,t)
        pause(0.15)
    end

end

function plot_sol(U,x,y,t)
    labelfontsize = 18;
```

```
    Lx = length(x);
    Ly = length(y);
    u = reshape(U, [Lx Ly]);
    figure(1)

    surf(x(2:Ly-1),y(2:Lx-1),u(2:Lx-1,2:Ly-1)')

    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel('$y$','interpreter','latex','fontsize',labelfontsize)
    zlabel(append('$u(x, y, t = ',num2str(t),')$'),'interpreter','
   latex','fontsize',labelfontsize)
    figName = create_figName(t);
    saveas(gcf,figName,'png')
end

function figName = create_figName(t)
    if t == 0
        exponent = 0;
        base = 00;
    else
        exponent = -2;
        base = t*100;
    end

    path = '../figures/';

    if base > 9
        figName = append(path,'p2_u_t',num2str(base),'e',num2str(
   exponent));
    else
        figName = append(path,'p2_u_t0',num2str(base),'e',num2str(
   exponent));
    end
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 $x \in [\pi, pi]$ 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 orthonormal basis functions $\phi_n(x)$ given,

$$\psi(x,t) \approx \psi_N(x,t) = \sum_{n=-N}^{N} c_n(t)\phi_n(x) = \frac{1}{\sqrt{2\pi}}\sum_{n=-N}^{N} c_n(t)e^{inx}$$

Define

$$\begin{aligned}
F(\psi_N) &:= \frac{i}{2}\frac{\partial^2}{\partial x^2}\psi_N(x,t) - iV(x)\psi_N(x,t) \\
&= \frac{1}{\sqrt{2\pi}}\frac{i}{2}\frac{\partial^2}{\partial x^2}\sum_{n=-N}^{N} c_n(t)e^{inx} - \frac{1}{\sqrt{2\pi}}iV(x)\sum_{n=-N}^{N} c_n(t)e^{inx} \\
&= -\frac{1}{\sqrt{2\pi}}\frac{i}{2}\sum_{n=-N}^{N} c_n(t)n^2 e^{inx} - \frac{1}{\sqrt{2\pi}}iV(x)\sum_{n=-N}^{N} c_n(t)e^{inx} \\
&= \frac{-i}{\sqrt{2\pi}}\sum_{n=-N}^{N}\left[c_n(t)\left(V(x) + \frac{n^2}{2}\right)e^{inx}\right].
\end{aligned}$$

Further,

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

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

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

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

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

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

$$
\begin{aligned}
I^V_{m,n} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} V(x) e^{i(n-m)x} dx \\
&= \frac{1}{2\pi} \int_{0}^{\pi} e^{i(n-m)x} dx,
\end{aligned}
$$

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

- For $n = m$:

$$
\begin{aligned}
I^V_{m,n} &= \frac{1}{2\pi} \int_{0}^{\pi} e^{i(n-m)x} dx \\
&= \frac{1}{2\pi} \int_{0}^{\pi} dx \\
&= \frac{1}{2}.
\end{aligned}
$$

- For $n \neq m$:

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

Problem 4

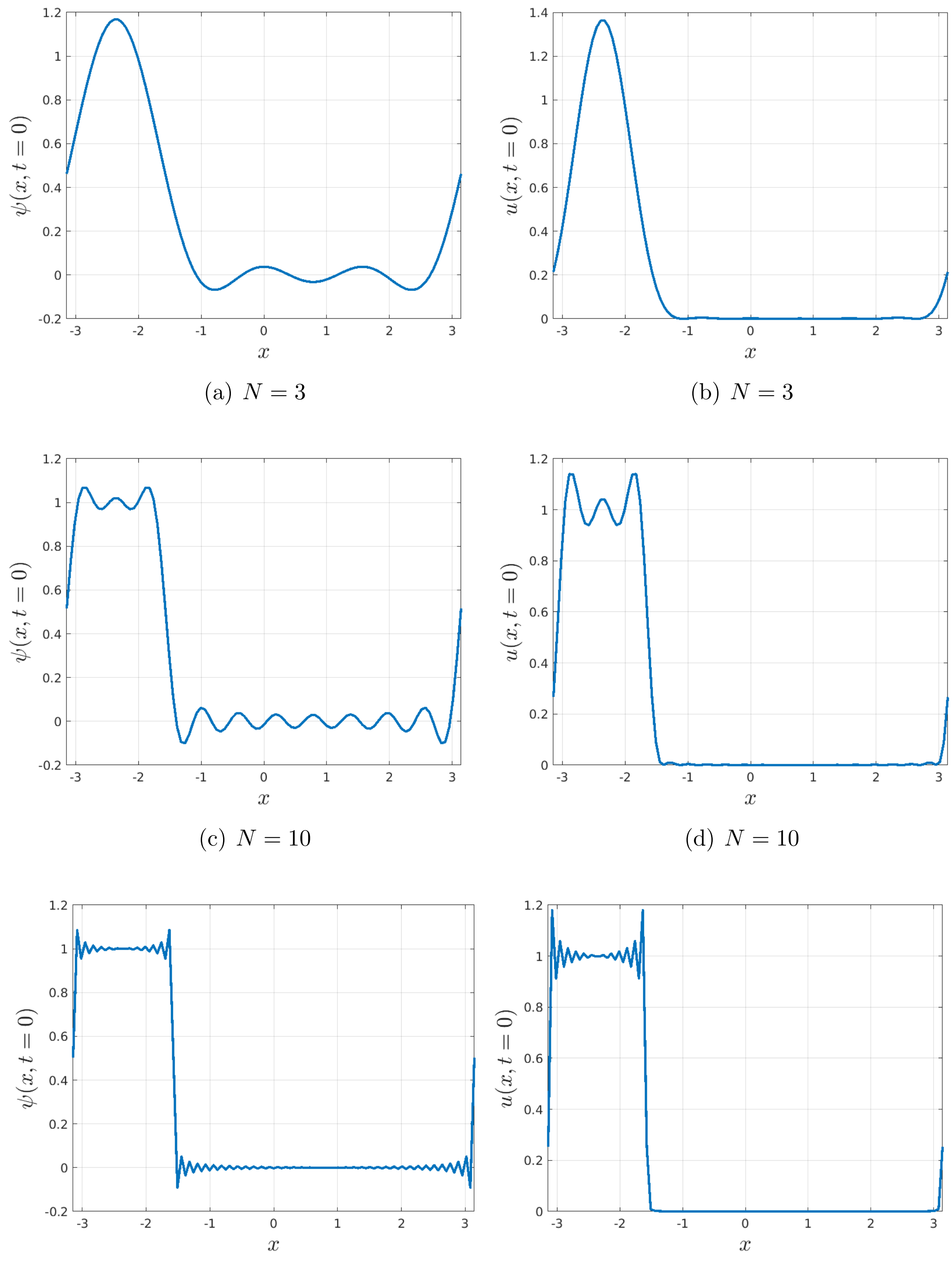
1. Solve the SE for $0 < t < 1$ for $N = 3, 10, 50$ with a time step Δt such that
$$\Delta t||A||^2 = \Delta t\sqrt{\rho(A^H A)} = 1$$
holds. Here ρ is the spectral radius $\rho(A^H A) = max_n|\lambda_n|$ with λ_n the eigenvalues of $A^H A$ and A^H the hermitian of the complex matrix A: $A^H = (A^T)^*$. 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 Cranck - Nicholson scheme for the time discretization.

Solution: From the previous problem, recover the equation
$$\overrightarrow{c}'(t) = A\overrightarrow{c}(t),$$
and apply the Crank-Nicholson scheme,
$$\frac{\overrightarrow{c}^{k+1} - \overrightarrow{c}^k}{\Delta t} = A\frac{1}{2}\left(\overrightarrow{c}^{k+1} + \overrightarrow{c}^k\right),$$
which yields
$$\overrightarrow{c}^{k+1} - A\frac{\Delta t}{2}\overrightarrow{c}^{k+1} = \overrightarrow{c}^k + A\frac{\Delta t}{2}\overrightarrow{c}^k.$$
Solving for $\overrightarrow{c}^{k+1}$,
$$\begin{aligned}\overrightarrow{c}^{k+1} &= \left(\mathbb{I} - A\frac{\Delta t}{2}\right)^{-1}\left(\mathbb{I} + A\frac{\Delta t}{2}\right)\overrightarrow{c}^k \\ &= M\overrightarrow{c}^k.\end{aligned}$$
Hence, we can find the solution at any time step k by
$$\overrightarrow{c}^k = M^k\overrightarrow{c}^0.$$
Note that we use the Crank-Nicholson scheme on the *coefficients* of the expansion. Hence, we need to calculate the coefficients of the initial condition,
$$\begin{aligned}c_n(0) = \left\langle\phi_n, \psi^I(x)\right\rangle &= \frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi} e^{-inx}\psi^I(x)dx \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\pi}^{-\pi/2} e^{-inx}dx \\ &= \frac{i}{n\sqrt{2\pi}}e^{in\pi/2}\left(1 - e^{in\pi/2}\right), \quad \text{for} \quad n \neq 0,\end{aligned}$$

and,

$$
\begin{aligned}
c_0(0) = \left\langle \phi_0, \psi^I(x) \right\rangle &= \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \psi^I(x) dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{-\pi/2} dx \\
&= \sqrt{\frac{\pi}{8}}.
\end{aligned}
$$

In the following figures, we can see the effect of the number of terms, N, used in the expansion. In the first figure we can see the dramatic effect that N has on the accuracy of the approximation of the initial condition. The remaining figures show the evolution with time of the solution ψ and the probability density u for each choice of N. Find the MATLAB code at the end.



(a) $N = 3$ (b) $N = 3$

(c) $N = 10$ (d) $N = 10$

(e) $N = 50$ (f) $N = 50$

Figure 2: Initial condition of the wave function (left) and probability density function (right) for different number of terms in the approximation.

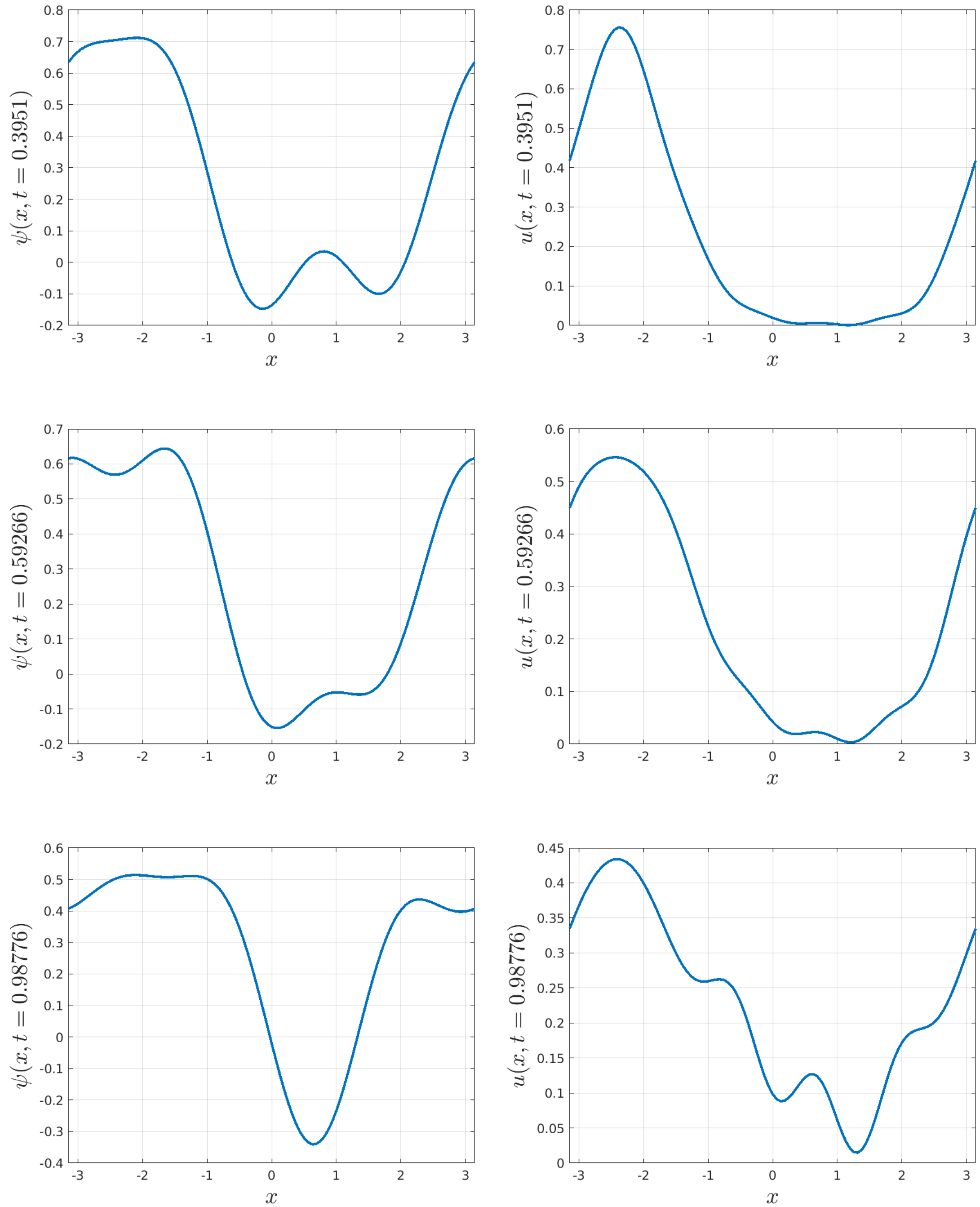


Figure 3: Evolution with time ot the wave function ψ (left) and probability density function u (right) for $N = 3$.

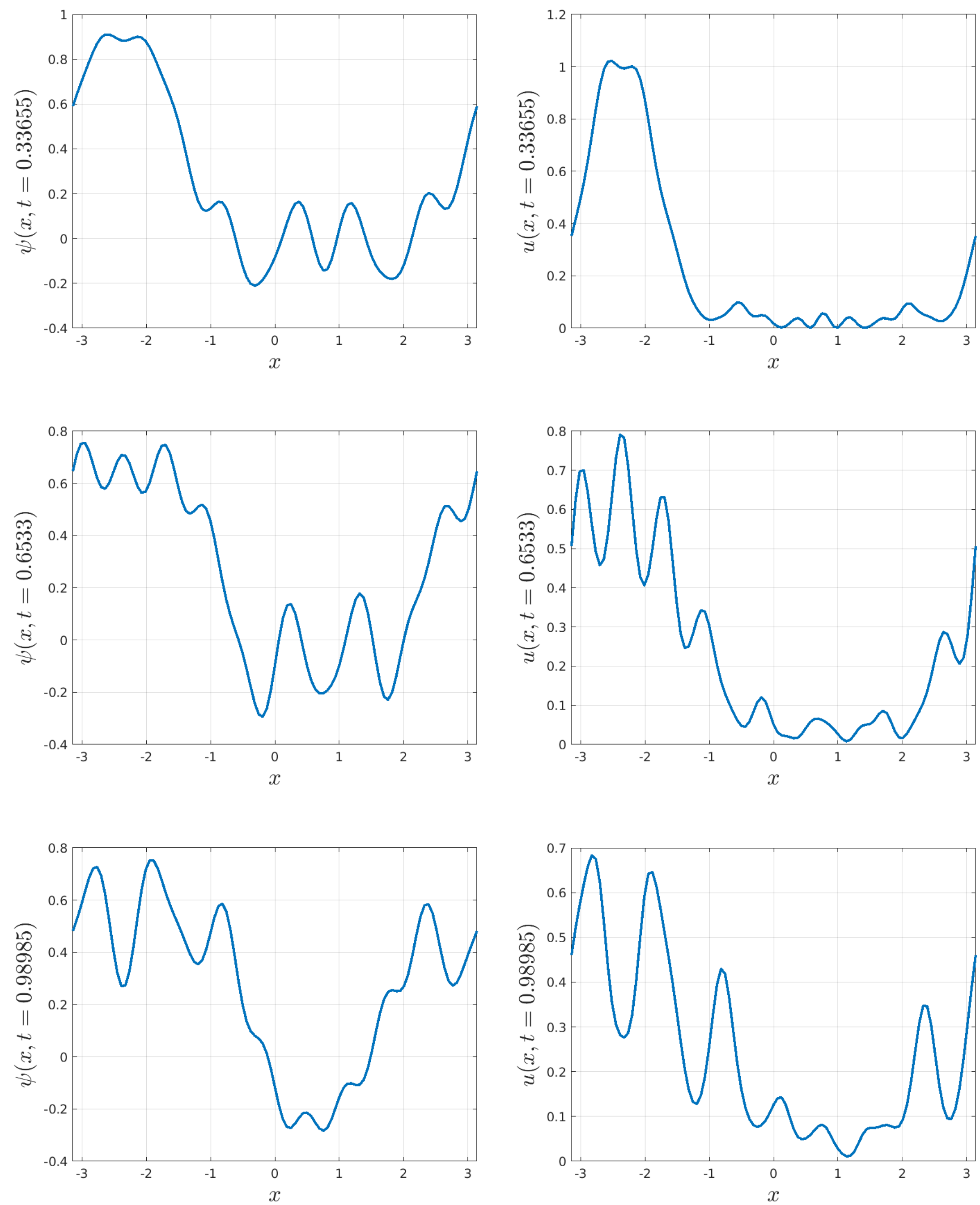


Figure 4: Evolution with time ot the wave function ψ (left) and probability density function u (right) for $N = 10$.

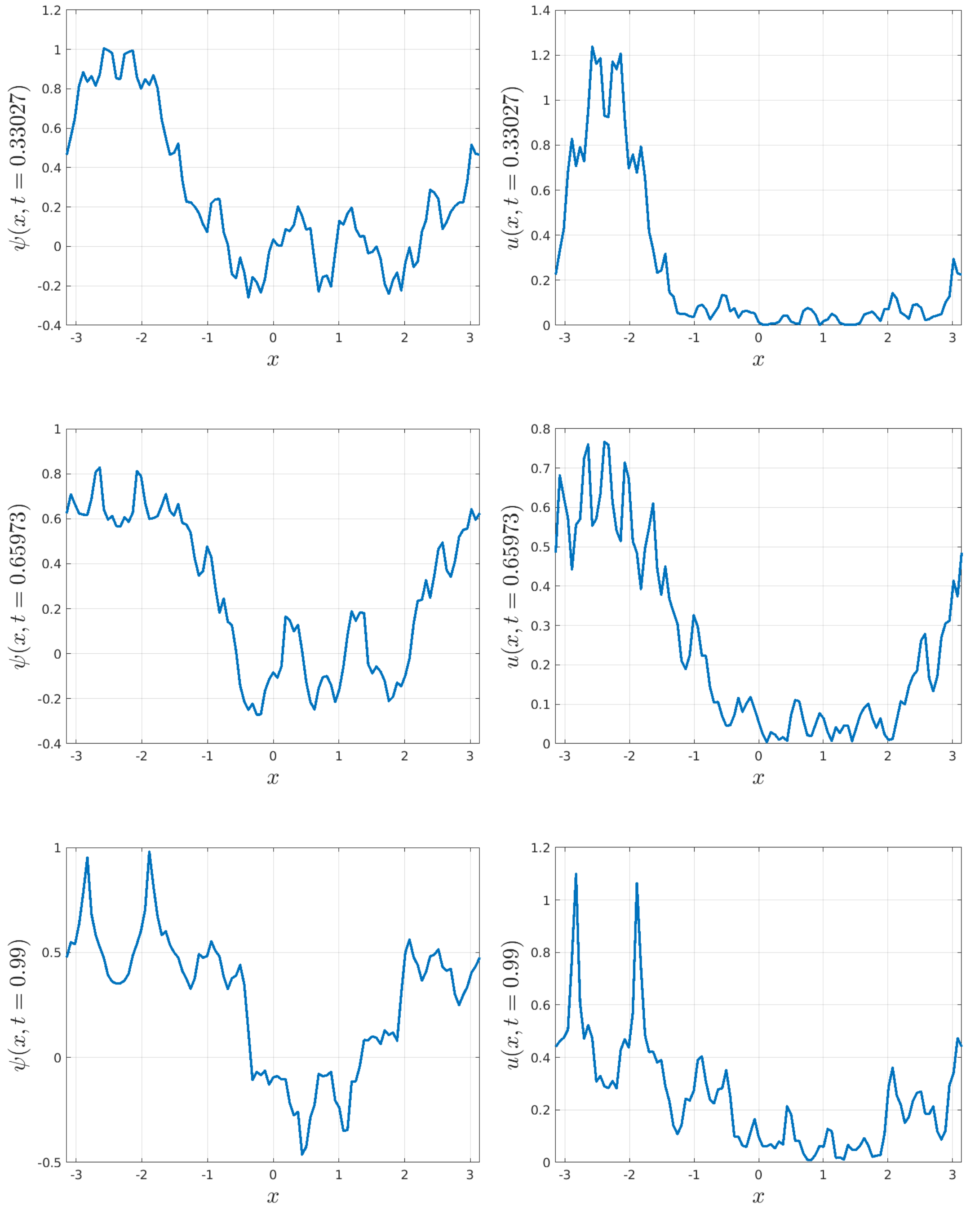


Figure 5: Evolution with time ot the wave function ψ (left) and probability density function u (right) for $N = 50$.

```matlab
clear all, close all, clc

N = 50; % Number of terms in expansion
n = (-N:N)'; % (2N+1, 1)

% Define vector x
Nx = 100; % Number of grid steps
x = linspace(-pi,pi,Nx+1); % (Nx+1, 1)

% Construct matrix A
A = calculate_A(n,N);

% Calculate dt
dt = calculate_dt(A);

% Define useful matrices
M1 = eye(size(A)) - 0.5*dt*A;
M2 = eye(size(A)) + 0.5*dt*A;
M = M1\M2; % As in c(k+1) = M c(k)

% Define initial coefficients for initial condition
c0 = calculate_initial_condition(n);
psi = calculate_wave_function(c0,n,x);
u = calculate_density(psi);
plot_psi(psi,x,0,0,N)
plot_u(u,x,0,0,N)

%% Find the time-steps to plot
T = 1;
t=0:dt:T;
t_plots = [0.33 0.66 0.99]; % Times to plot
tstep_plots = zeros(3,1);
for i=1:length(t_plots)
    [minValue, minIdx] = min(abs(t-t_plots(i)));
    tstep_plots(i) = minIdx;
end
tstep_plots
t(tstep_plots)
keyboard
%% Compute solutions and plot
for idx=1:length(tstep_plots)
    tstep = tstep_plots(idx);
    c = M^tstep*c0;
    t = (tstep-1)*dt;
    psi = calculate_wave_function(c,n,x);
    u = calculate_density(psi);
    plot_psi(psi,x,t,idx,N)
    plot_u(u,x,t,idx,N)
end

function A = calculate_A(n,N)
    m = n;
```

```
    I = 1i * (1 - (-1).^(n'-m)) ./ (2 * pi * (n'-m));
    I(1:2*N+2:(2*N+1)^2) = 1/2; % Change diagonal values
    A = -1i * (I + diag(n.^2/2));
end

function dt = calculate_dt(A)
    spectral_radius = max(eig(A'*A));
    dt = 1/ sqrt(spectral_radius);
end

function c0 = calculate_initial_condition(n)
    c0 = 1i * exp(1i*n*pi/2) .* ( 1 - exp(1i*n*pi/2) ) ./ (sqrt(2*pi) * n);
    c0(n == 0) = sqrt(pi/8);
end

function psi = calculate_wave_function(c,n,x)
    psi = transpose(c) * exp(1i * n * x)/sqrt(2*pi);
end

function u = calculate_density(psi)
    u = abs(psi).^2;
end

function plot_psi(psi,x,t,idx,N)
    labelfontsize = 18;

    figure(1)
    plot(x,psi,'linewidth',2)
    xlim([-pi pi])
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel(append('$\psi(x, t = ',num2str(t),')$'),'interpreter','latex','fontsize',labelfontsize)
    grid on
    figName = create_figName('psi',idx,N);
    saveas(gcf,figName,'png')
end

function plot_u(u,x,t,idx,N)
    labelfontsize = 18;

    figure(2)
    plot(x,u,'linewidth',2)
    xlim([-pi pi])
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel(append('$u(x, t = ',num2str(t),')$'),'interpreter','latex','fontsize',labelfontsize)
    grid on
    figName = create_figName('u',idx,N);
    saveas(gcf,figName,'png')
end
```

```
function figName = create_figName(name,idx,N)
    path = '../figures/';
  figName = append(path,'p4_',name,'_N',num2str(N),'_sample',num2str
    (idx));
end
```