Prof. Mark D Shattuck
Physics 39907 Computational Physics
November 13, 2025

Final Exam

Question 1. *Finite Element Method (FEM):* FEM can be used to find the solution $u(x)$ to differential equations like those based on:

$$u(x) \xrightarrow{A=\frac{d}{dx}} e(x) = \frac{d}{dx}u(x) \xrightarrow{C(x)} w(x) = C(x)e(x) \xrightarrow{A^T=-\frac{d}{dx}} -\frac{d}{dx}w(x) = f(x),$$

which gives a strong form:

$$-\frac{d}{dx}w(x) = f(x) \tag{1}$$

$$-\frac{d}{dx}\left(C(x)e(x)\right) = f(x) \tag{2}$$

$$-\frac{d}{dx}\left(C(x)\frac{d}{dx}u(x)\right) = f(x) \quad \text{plus B.C.} \tag{3}$$

For FEM convert the strong form [3] to the weak form by taking the inner product of the strong form with a test function $v(x)$ and applying integration by parts:

$$\int_a^b \frac{du(x)}{dx}v(x)\ dx = [u(x)v(x)]_a^b - \int_a^b u(x)\frac{dv(x)}{dx}\ dx$$

The inner product on the range $[a, b]$ of $u(x)$ and $v(x)$ is

$$(u(x), v(x)) = \int_a^b u(x)v(x)\ dx.$$

Then the weak form of [3] is:

$$\left(-\frac{d}{dx}\left(C(x)\frac{d}{dx}u(x)\right), v(x)\right) = (f(x), v(x)), \tag{4}$$

$$\int_a^b -\frac{d}{dx}\left(C(x)\frac{du(x)}{dx}\right)v(x)\ dx = \int_a^b f(x)v(x)\ dx, \tag{5}$$

$$\int_a^b C(x)\frac{du(x)}{dx}\frac{dv(x)}{dx}\ dx - \left[C(x)\frac{u(x)}{dx}v(x)\right]_a^b = \int_a^b f(x)v(x)\ dx, \tag{6}$$

$$\int_a^b C(x)u'(x)v'(x)\ dx - [C(x)u'(x)v(x)]_a^b = \int_a^b f(x)v(x)\ dx. \tag{7}$$

(1) Given the strong form:

$$-u''(x) = \delta(x - a) \quad \text{with } u'(0) = 0,\ u(1) = 0,\ 0 < a < 1,$$

show that the weak form is:

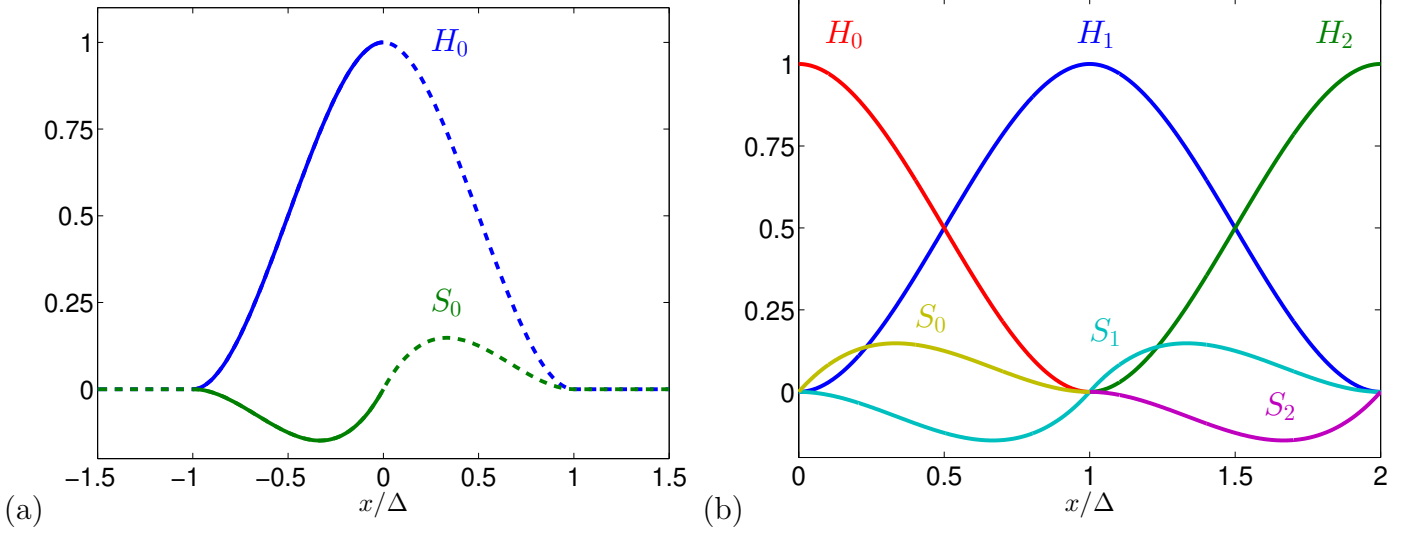$$\int_0^1 u'(x)v'(x)\ dx = v(a). \tag{8}$$

FIGURE 1. Piece-wise Cubic functions (a) at node 0 and (b) all functions that overlap H_1.

(2) In FEM, the solution $u(x)$ is approximated by:

$$u(x) = \sum_{k=0}^{K} u_k \phi_k(x). \tag{9}$$

For this problem we will use two piece-wise cubic functions centered at each node located at $x_n = n\Delta$: a (H)eight function $H_n(x)$ and a (S)lope function $S_n(x)$ and corresponding coefficients u_n^H and u_n^S. Figure 1(a) shows $H_0(x)$ and $S_0(x)$ centered at node 0. The functions are zero and have zero slope at and beyond adjacent nodes at ± 1. At the central node 0, $H_0(0)$ has height 1 and slope 0, but $S_0(0)$ has slope 1 and height 0, so that $u_n^H H_n(x) + u_n^S S_n(x)$ has height u_n^H and slope u_n^S at node n. Using the symmetries, $H_0(x) = H_0(-x)$ and $S_0(x) = -S_0(-x)$, we can define them in terms local functions $H(x/\Delta)$ and $S(x/\Delta)$ on the interval $[-\Delta/\Delta, 0] = [-1, 0]$, shown as the solid lines in figure 1(a) as follows:

$$H_0(x) = H_0(x; \Delta) = \begin{cases} 0 & x/\Delta \leq -1 \\ H(x/\Delta) & -1 \leq x/\Delta \leq 0 \\ H(-x/\Delta) & 0 \leq x/\Delta \leq 1 \\ 0 & x/\Delta \geq 1 \end{cases} \tag{10}$$

and

$$S_0(x) = S_0(x; \Delta) = \begin{cases} 0 & x/\Delta \leq -1 \\ S(x/\Delta) & -1 \leq x/\Delta \leq 0 \\ -S(-x/\Delta) & 0 \leq x/\Delta \leq 1 \\ 0 & x/\Delta \geq 1 \end{cases} \tag{11}$$

H and S are defined in local grid coordinates x/Δ. These functions can be shifted to other nodes as shown in figure 1(b) using this equation $H_n(x) = H_0((x/\Delta - n)\Delta)$ and $S_n(x) = S_0((x/\Delta - n)\Delta)$. All of the functions $H_n(x)$ and $S_n(x)$ are shown in figure 1(b) for the interval $[0, 2\Delta]$. This represents all of the functions that overlap $H_1(x)$ and $S_1(x)$.

Find a cubic function $H(x)$ with the following properties:
  (a) $H(x)$ is a cubic. For example, $H(x) = s(x - a)(x - b)(x - c)$.
  (b) The derivative $H'(-1) = 0$ and $H'(0) = 0$.
  (c) $H(-1) = 0$ and $H(0) = 1$.
Show that the cubic function $S(x) = x(x + 1)^2$ has the following properties:
  (a) $S(x)$ is a cubic.
  (b) The derivative $S'(-1) = 0$ and $S'(0) = 1$.
  (c) $S(-1) = 0$ and $S(0) = 0$.
(3) Using the approximation above, the solution for nodes 0-N is

$$u(x) = \sum_{n=0}^{N} u_n^S S_n(x) + u_n^H H_n(x), \qquad [12]$$

and

$$u'(x) = \sum_{n=0}^{N} u_n^S S_n'(x) + u_n^H H_n'(x). \qquad [13]$$

From this equation or figure 1 find the value of $u(\Delta)$, $u(\Delta/2)$, and $u'(\Delta)$ in terms of u_n^S and u_n^H.
(4) Show that [12] can be written in matrix form:

$$u(x) = \begin{bmatrix} u_0^S & u_0^H & \cdots & u_N^S & u_N^H \end{bmatrix} \begin{bmatrix} S_0(x) \\ H_0(x) \\ \vdots \\ S_N(x) \\ H_N(x) \end{bmatrix} = u^T \phi(x)$$

and find a similar equation for $u'(x)$ from [13]. What is the shape (size) of u and ϕ?
(5) Plug [12] into the weak form [8] for

$$v_k(x) = \begin{bmatrix} v_0(x) \\ \vdots \\ v_{2N+1}(x) \end{bmatrix} = \begin{bmatrix} S_0(x) \\ H_0(x) \\ \vdots \\ S_N(x) \\ H_N(x) \end{bmatrix} = \phi(x)$$

to show that

$$\sum_{n=0}^{N} \int_0^1 (u_n^S S_n'(x) + u_n^H H_n'(x)) v_k'(x) \, dx = v_k(a) \qquad [14]$$

$$\left(\int_0^1 \phi'(x)\phi'(x)^T \, dx \right) u = \phi(a) \qquad [15]$$

$$Ku = f. \qquad [16]$$

What is the shape of K? Is K symmetric? Why? Why not?
(6) Set up [16] to solve [3] with $a = 3/8$, on a grid with 5 nodes and 4 intervals $x = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \end{bmatrix}^T \Delta$, $\Delta = 1/4$ by follow these steps:

(a) Fill in the missing elements in f for $N = 5$:

$$f = \begin{bmatrix} S_0(3/8) \\ H_0(3/8) \\ S_1(3/8) \\ H_1(3/8) \\ S_2(3/8) \\ H_2(3/8) \\ S_3(3/8) \\ H_3(3/8) \\ S_4(3/8) \\ H_4(3/8) \end{bmatrix} = \begin{bmatrix} 0 \\ ??? \\ S_0((3/8/\Delta - 1)\Delta) = S_0(1/2\Delta) = -S(-1/2) = -(-1/2)(-1/2+1)^2 = 1/8 \\ ??? \\ ??? \\ 1/2 \\ 0 \\ ??? \\ 0 \\ 0 \end{bmatrix}$$

(b) Find the functions S_0', H_0', S_1', H_1' in the interval $[0, 1]$. From figure 1(b) notice that interval from $[0, 1]$ is repeated in the interval $[1, 2]$, but with $H_0 \to H_1$, $H_1 \to H_2$, $S_0 \to S_1$, $S_1 \to S_2$. Therefore all of the terms in K can be constructed from just the overlaps in the interval $[0, 1]$. Fill in the missing function in this vector:

$$\phi_{loc} = \begin{bmatrix} S_0'(x) \\ H_0'(x) \\ S_1'(x) \\ H_1'(x) \end{bmatrix} = \begin{bmatrix} 3x^2 - 4x + 1 \\ 6x(x - 1) \\ x(3x - 2) \\ ??? \end{bmatrix}$$

(c) Evaluate [15] using the local ϕ_{loc}. There are 4 local function in each unit interval. Fill in the missing integrals in

$$K_{loc} = \int_0^1 \phi_{loc} \phi_{loc}^T \, dx = \int_0^1 \begin{bmatrix} S_0'(x) \\ H_0'(x) \\ S_1'(x) \\ H_1'(x) \end{bmatrix} \begin{bmatrix} S_0'(x) & H_0'(x) & S_1'(x) & H_1'(x) \end{bmatrix} \, dx$$

$$= \begin{bmatrix} \int_0^1 S_0'(x)S_0'(x)\,dx & \int_0^1 S_0'(x)H_0'(x)\,dx & ??? & ??? \\ \int_0^1 H_0'(x)S_0'(x)\,dx & \int_0^1 H_0'(x)H_0'(x)\,dx & ??? & ??? \\ ??? & ??? & ??? & ??? \\ ??? & ??? & ??? & ??? \end{bmatrix}$$

$$= \frac{1}{30} \begin{bmatrix} 30\int_0^1 (3x-1)^2(x-1)^2\,dx = 4 & ??? & ??? & ??? \\ 30\int_0^1 6x(3x-1)(x-1)^2\,dx = 3 & ??? & ??? & ??? \\ -1 & 3 & ??? & ??? \\ -3 & -36 & -3 & ??? \end{bmatrix}$$

$$= \frac{1}{30} \begin{bmatrix} 4 & ??? & ??? & ??? \\ 3 & ??? & ??? & ??? \\ -1 & 3 & ??? & ??? \\ -3 & -36 & -3 & ??? \end{bmatrix}$$
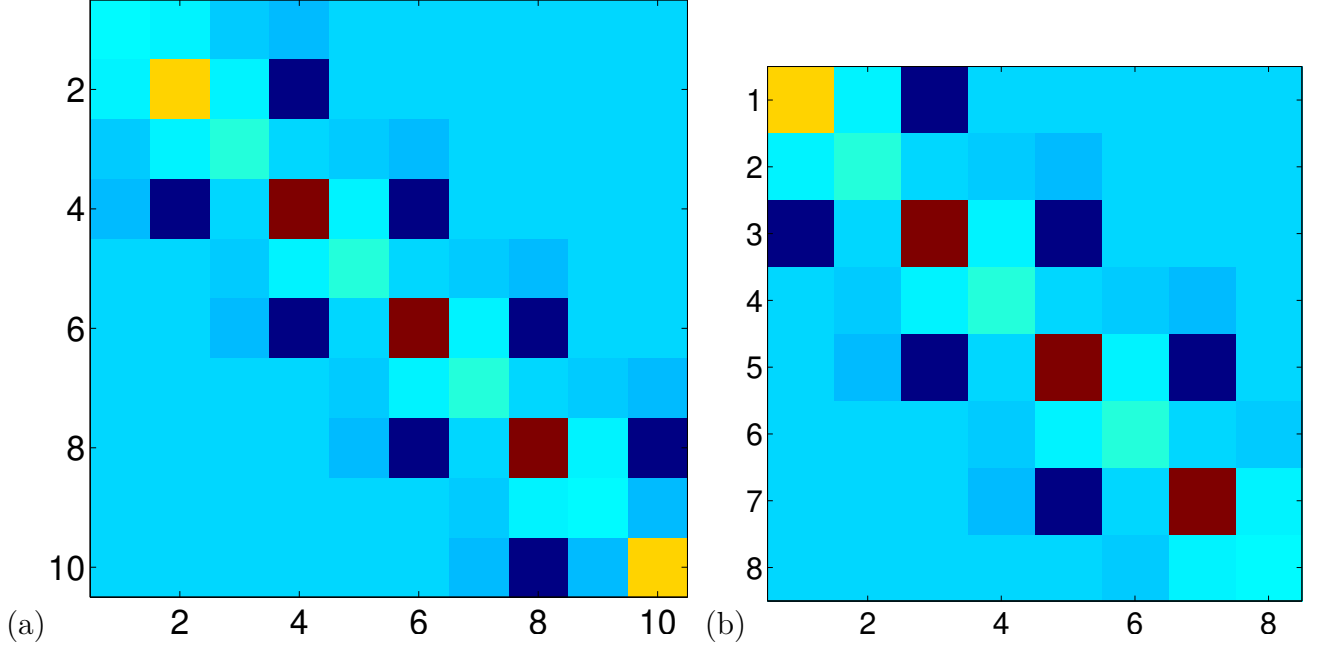
FIGURE 2. Global K.

(d) Use the following equation to build the global K from K_{loc}. The local matrix is shifted by 2 in each direction, then all of them are added for each unit in the grid.

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & 0 & 0 & 0 & \ldots & 0 \\ k_{21} & k_{22} & k_{23} & k_{24} & 0 & 0 & 0 & \ldots & 0 \\ k_{31} & k_{32} & k_{33} & k_{34} & 0 & 0 & 0 & \ldots & 0 \\ k_{41} & k_{42} & k_{43} & k_{44} & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & k_{11} & k_{12} & k_{13} & k_{14} & 0 & \ldots & 0 \\ 0 & 0 & k_{21} & k_{22} & k_{23} & k_{24} & 0 & \ldots & 0 \\ 0 & 0 & k_{31} & k_{32} & k_{33} & k_{34} & 0 & \ldots & 0 \\ 0 & 0 & k_{41} & k_{42} & k_{43} & k_{44} & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} + \ldots$$

$$+ \begin{bmatrix} 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 & k_{11} & k_{12} & k_{13} & k_{14} \\ 0 & \ldots & 0 & 0 & 0 & k_{21} & k_{22} & k_{23} & k_{24} \\ 0 & \ldots & 0 & 0 & 0 & k_{31} & k_{32} & k_{33} & k_{34} \\ 0 & \ldots & 0 & 0 & 0 & k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}$$

The properly assembled K is shown in figure 2(a). The figure was made using the following MATLAB command:
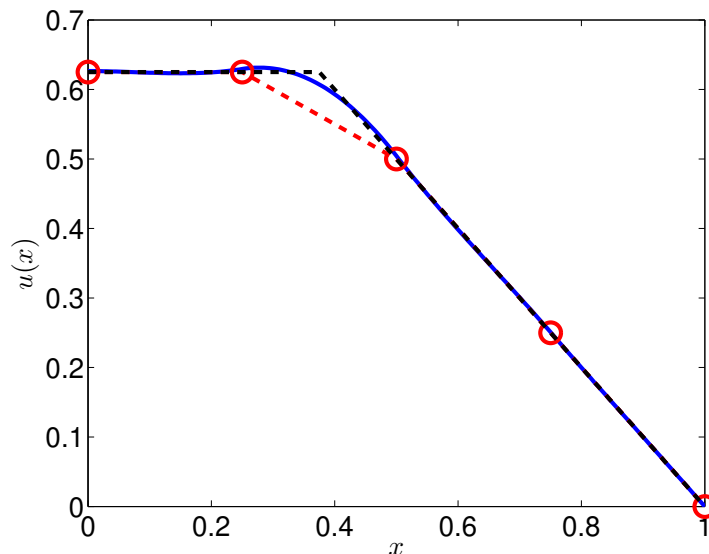
```
imagesc(K); axis('image'); colormap(jet(256));
```

FIGURE 3. Comparison of FEM, finite differences, and exact result.

(e) Apply the boundary conditions. Eliminate any S_n or H_n that are determined by the boundary conditions and trim the global K. The final K is shown in figure 2(b).

(7) Use the K and f defined above to solve for u and report the values.

(8) Use the following MATLAB script (FEM000.m) to plot the results:

```matlab
1  N=5;        % Number of nodes
2  a=3/8;      % Location of forcing delta function
3
4  dx=1/(N-1);   % grid spacing \Delta
5  K=???; % fill in the values for K
6  f=???; % fill in the values for f
7
8  u=???; % solve for u. note: include any boundary values as well.
9
10 u2=???; % solve the same problem using finite differences e.g., free-fixed (T)
11
12 x=0:dx/50:1;   % locations to find U(x)
13
14 U=evalFEM(x,u,dx); % calculate U(x) from x, u, and, dx
15
16 ue=(1-x).*(x>a)+(1-a).*(x<=a);   % exact solution
17
18 % plot results
19 h=plot(x,U,(0:N-1)*dx,u2,'ro--',x,ue,'k--');
20
21 % make it pretty
22 set(h,'linewidth',3,'markersize',15);
23 set(gca,'fontsize',20);
24 xlabel('$x$','interp','latex');
25 ylabel('$u(x)$','interp','latex');
```

In the script you will need to provide K and f, and code to calculate the resulting coefficients u. The MATLAB function evalFEM() used to evaluate equation [12] is here evalFEM.m and listed below. It uses absolute values to express [10] and [11] more compactly in local coordinates $\nu = x/\Delta$:

$$S_0(\nu) = \begin{cases} \nu(|\nu| - 1)^2 & |\nu| < 1 \\ 0 & else \end{cases}$$

$$H_0(\nu) = \begin{cases} (2|\nu| + 1)(|\nu| - 1)^2 & |\nu| < 1 \\ 0 & else \end{cases}$$

You will need to supply code to calculate the same solution using 5-point finite differences. The exact solution:

$$u_e(x) = \begin{cases} (1 - x)/4 & a \geq x \leq 1 \\ (1 - a)/4 & 0 \geq x \leq a \end{cases}$$

Use the code above or your own code to plot the FEM, FD, and exact solution.

```matlab
1  function [U,S,H]=evalFEM(x,u,dx,H,S)
2  % evalFEM <Find value of C1 cubic FEM>
3  % Usage:: [U,S,H]=evalFEM(x,u,dx[1],...
4  %    H[@(x) (2*abs(x)+1).*(abs(x)-1).^2.*(abs(x)<1)],...
5  %    S[@(x) x.*(abs(x)-1).^2.*(abs(x)<1)])
6  %
7
8  % revision history:
9  % 12/10/2023 Mark D. Shattuck <mds> evalFEM.m
10
11 %% Parse Input
12 if(~exist('dx','var') || isempty(dx))
13     dx=1;
14 end
15
16 if(~exist('H','var') || isempty(H))
17     H=@(x) (2*abs(x)+1).*(abs(x)-1).^2.*(abs(x)<1);
18 end
19
20 if(~exist('S','var') || isempty(S))
21     S=@(x) x.*(abs(x)-1).^2.*(abs(x)<1);
22 end
23
24 %% Main
25 N=length(u)/2;
26 U=0;
27 for n=0:N-1;
28    U=U+u(2*n+1)*S(x/dx-n)+u(2*n+2)*H(x/dx-n);
29 end;
```

Question 2. *Partial Differential Equation PDE:* A PDE is a differential equation which depends on derivatives of more than one variable. In this problem, we will solve a modified 2D Cahn–Hilliard equation on periodic boundary conditions:

$$\frac{\partial c}{\partial t} = \nabla^2 \mu \tag{1}$$

$$\mu = W(c) - \gamma \nabla^2 c \tag{2}$$

$$W(c) = (c-1)c(c-1/2) \tag{3}$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{4}$$

and $c = c(x, y, t)$ is a function of space and time. This equation determines the concentration $c(x, y, t)$ of one fluid mixed with another immiscible fluid, like oil and water. A concentration of 1 at position (x, y) and time t means all of one fluid. The concentration of the second fluid is $1 - c(x, y, t)$. If the fluids start mixed, they will demix over time.

(1) Convert the equations to 1 dimension, by eliminating y. To get an equation of the form:

$$\frac{\partial c(x, t)}{\partial t} = A(c(x, t)).$$

The function A will depend on c and its x-derivatives.

(2) Create a MATLAB script (outlined below) to begin solving these equation. You will need the constant `gam=3e-5`, the grid size `Nx=128`, a domain of size `Lx=1;`, and a time step of `dt=1e-6`. From these calculate the grid spacing `dx=???`.

(3) We will use first-order forward Euler integration to solve the equation in time. Discretize the equation in time and write the first order approximation for

$$\frac{\partial c(x, t)}{\partial t} \approx \frac{c_{n+1}(x) - c_n(x)}{\Delta t} = A(c_n(x)),$$

and solve for $c_{n+1}(x)$. This represents our update rule. What is $c_n(x)$ in terms of $c(x, t)$?

(4) To update $c_n(x)$ we need an initial condition. Add a variable `c` to your code to represent $c_n(x)$ and set the initial condition to get a random 0 or 1 at each location. A good way to get random 1's and 0's is with `rand(3,1)>1/2`. This will give a 3×1 column vector of random 1's and 0's. `c` should be a column vector of size `[Nx,1]`.

(5) To evaluate $A(c(x))$ second derivatives are needed. If we use a discrete representation of $c_k = c(k\Delta x)$, the second derivative is:

$$c''(k\Delta x) \approx \frac{c_{k-1} - 2c_k + c_{k+1}}{(\Delta x)^2}$$

With periodic boundary conditions the matrix version is `Dxx*c`, where

```
Dxx=toeplitz([-2 1 0 ??? 0 1])/dx/dx;.
```

Add this to your code and replace the `0 ??? 0` so that `Dxx*c` works for any size `Nx` vector `c`.

(6) Test `Dxx` on `sin(2*pi*x)`, where `x` is size `[Nx,1]` and goes from 0 to `1-1/Nx`. When it is working `Dxx*sin(2*pi*x)` should be approximately `-(2*pi)^2*sin(2*pi*x)`, since $(\sin(ax))'' = -a^2 \sin(x)$.

(7) Putting it all together. Add a loop to your code that will use the update rule above to move forward by steps of `dt`. The code will calculate $A(c)$ then update c then repeat. Add an integration total time `TT=.05;` to your code. Calculate the integer number of time steps `Nt` needed to reach `TT`

i.e., `Nt*dt` is approximately `TT`. Here is my version with some blanks. It includes code to plot the solution, comments, and code to save the result which you should add to your code.

```matlab
 1  %% 1D Cahn-Hillard Simulator
 2  % <CH1d.m> Mark D. Shattuck 12/10/2023
 3
 4  % revision history:
 5  % 12/10/2023 Mark D. Shattuck <mds> CH1d.m
 6  %
 7  % 12/10/2023 mds set up for PHYS 339 final
 8  %
 9  %% Experimental Parameters
10  gam=.00003;   % control parameter
11
12  Nx=128;        % Number of grid points on
13  Lx=1;          % Size of container
14
15  TT=.05;        % Total simulation time
16
17  %% Simulation parameters
18  dt=1e-6;
19
20  %% Calculated parameters
21  Nt=???;       % number of Time steps
22  dx=???;       % grid spacing
23  x=(0:Nx-1)'*dx;  % x-grid for plotting and testing
24
25  % 2nd derivative of a column vector
26  Dxx=toeplitz([-2 1 ???? 1])/dx/dx;
27  Dxx=sparse(Dxx);  % convert to sparse for speed
28
29  %% initial conditions
30  c=rand(Nx,1)>1/2;  % random initial condition
31
32  %% Save State
33  cs=zeros(Nx,Nt);  % save every time step
34
35  %% Main loop
36
37  for nt=1:Nt
38    mu=???;      % mu is function of c, Dxx, and gam
39    dc=Dxx*mu; % from equation [1]
40    c=c+???;     % update rule
41
42    % give feedback by plotting
43    if(rem(nt,fix(Nt/100))==0)
44      plot(x,c)
45      drawnow;
46      disp([nt/100 mean(c(:))]);
47    end
48
49    cs(:,nt)=c;  % save results
50  end
```
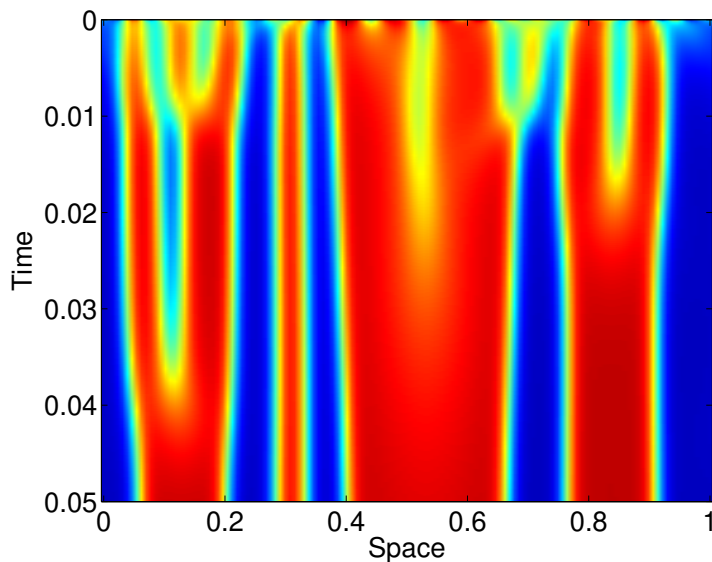
FIGURE 4. Space-time plot of 1D Cahn-Hillard equation.

When it is working use:

```
1  imagesc([0 Lx],[0 TT],cs');
2  xlabel('Space');
3  ylabel('Time');
4  colormap(jet(256));
```

to get a plot like figure 4. The red is one fluid and the blue is the second fluid. The red regions separate from the blue.

(8) Copy your 1D code to a new script and convert to 2D. There is not a lot to change. The main issue is the derivatives in y. If you convert c from a [Nx,1] matrix to a [Nx,Ny] matrix, then it turns out that multiplying c*Dyy' from the right by the transpose will take the derivative in the other direction, where Dyy is defined in analogy to Dxx. Second derivatives are symmetric Dyy=Dyy' so the transpose is not needed. Here is my version with missing parts:
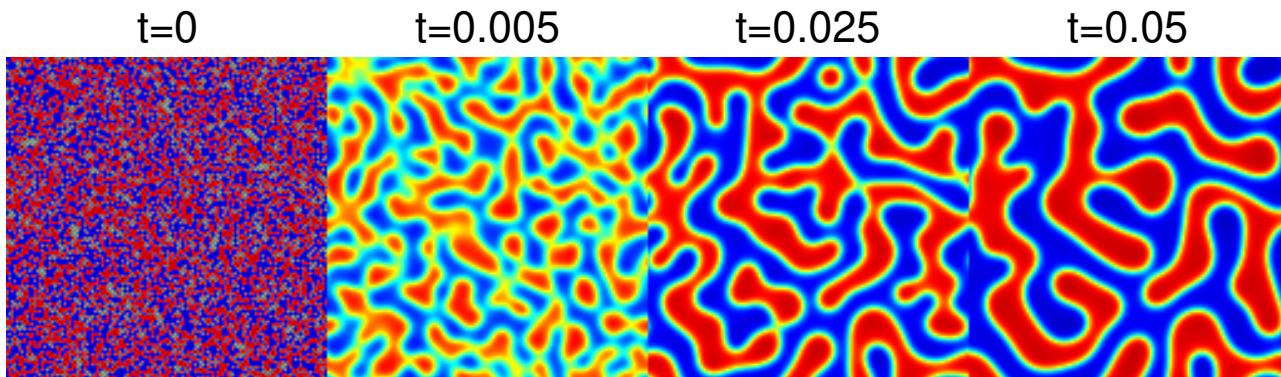

FIGURE 5. 2D evolution of Cahn-Hillard equation

```matlab
1  %% 2D Cahn-Hillard Simulator
2  % <CH1d.m> Mark D. Shattuck 12/10/2023
3
4  % revision history:
5  % 12/10/2023 Mark D. Shattuck <mds> CH1d.m
6  %
7  % 12/10/2023 mds set up for PHYS 339 final
8  % 12/14/2023 mds conver to 2D CH2d.m
9
10 %% Experimental Parameters
11 gam=.00003;  % control parameter
12
13 Nx=128;       % Number of grid points in x
14 Ny=128;       % Number of grid points in y
15 Lx=1;         % Size of container in x
16 Ly=1;         % Size of container in y
17
18 TT=.05;       % Total simulation time
19
20 %% Simulation parameters
21 dt=1e-6;
22
23 %% Calculated parameters
24 Nt=???;    % number of Time steps
25 dx=???;         % x-grid spacing
26 dy=???;         % y-grid spacing
27
28 % 2nd derivative of a matrix
29 Dxx=toeplitz([-2 1 ???? 1]/dx/dx);
30 Dxx=sparse(Dxx);  % convert to sparse for speed
31
32 Dyy=toeplitz([-2 1 ????? 1]/dy/dy);
33 Dyy=sparse(Dyy);  % convert to sparse for speed
34
35 %% initial conditions
36 c=????;  % random initial condition now (Nx,Ny)
37
38 %% Main loop
39 for nt=1:Nt
40   mu=????;              % mu is function of c, Dxx, Dyy, and gam
41   dc=Dxx*mu+mu*Dyy;   % from equation [1]
42   c=c+???;              % update rule
43
44   % give feedback by plotting
45   if(rem(nt,fix(Nt/200))==0)
46     imagesc([0 Ly],[0 Lx],c);  % now display current image
47     axis('image');
48     drawnow;
49     disp([nt/100 mean(c(:))]);
50   end
51 end
```

When it is working it will look like figure 5.

(9) Try changing some things a see what happens. Some examples:
   (a) Make `Ly` and/or `Ny` bigger or smaller.
   (b) Change the initial condition so that there are more or less 1's.
   (c) Changing the $1/2$ in $W(c)$ is interesting, $1/4$ or $3/4$.
   (d) What happens if `dt` is too big? How big can it be? Is the maximum `dt` effected by other parameters.
   (e) What does `gam` do?