

# Lecture 21: Basic time- & space- stepping schemes

- Partial Differential Equations (PDEs) require discretization in space as well as time.
- One way to solve PDEs is to discretize in space reducing the PDEs to a system of Ordinary Differential Equations (ODEs) in time.
- We can then advance the solutions in time using the time stepping schemes we've already learned in previous lectures.

For example, solve the diffusion equation (otherwise known as the heat equation):

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}$$

on a domain  $x \in [-L, L]$  with periodic boundary conditions  $u(t, -L) = u(t, L)$

We take a second order finite difference approximation for the second derivative of  $u(t, x)$ :

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(t, x + \Delta x) - 2u(t, x) + u(t, x - \Delta x)}{\Delta x^2}$$

If we arrange our unknown vector  $\mathbf{u}$  for each spatial node:  $-L : \Delta x : L - \Delta x$  as

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} u(t, -L) \\ u(t, -L + \Delta x) \\ \cdot \\ \cdot \\ \cdot \\ u(t, L - 2\Delta x) \\ u(t, L - \Delta x) \end{pmatrix}$$

Note: we do not include  $u(t, L)$  because  $u_{n+1} = u_1 = u(t, L)$  from the periodic boundary conditions.

We can then write the spatially discretized original PDE as an ODE system:

$$\frac{d\mathbf{u}}{dt} = \frac{\kappa}{\Delta x^2} \mathbf{A}\mathbf{u}$$

**A** is constructed from the difference approximation

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(t, x + \Delta x) - 2u(t, x) + u(t, x - \Delta x)}{\Delta x^2}$$

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ \cdot & 1 & -2 & 1 & \dots & 0 \\ \vdots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 1 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}$$

Note the 1 in the last column and first row, and the 1 in the first column and last row. These correspond to the periodic boundary conditions.

To implement this in Matlab using ode45, take for example the heat equation with  $\kappa = 1.2$  and domain with  $L = 1$ :

```
% define ODE rhs in a file rhs.m
function rhs=rhs(tspan,u,dummy,k,dx,A)
rhs=(k/dx^2)*A*u;

% build a vector of ones
n = 20;
e1=ones(n,1);

% construct spatial discretization matrix A
A=spdiags([e1 -2*e1 e1],[-1 0 1],n,n);

% periodic boundaries:
A(1,n)=1; A(n,1)=1;

% define the domain and time span
dx = (1-(-1))/n
x = (-1:dx:1-dx);
tspan = [0 10];
```

```
% initial conditions (gaussian)
u0 = exp(-x.^2);

% solve ODE in time
k = 1.2;
[t,y]=ode45('rhs',tspan,u0,[],k,dx,A);

plot(x,u0,'r*-',x,y(1,:),'g*-',x,y(end,:),'b*-');
legend('Initial condition', 'Solution at t = dt\_1',
'Solution at t = 10')

% plot solutions as a function of time at x = 0
figure(2)
set(gca, 'FontSize', 20);
I = find(x == 0);
plot(t,y(:,I),'b*-')
```

Solving a PDE with two spatial dimensions

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

on a domain  $(x, y) \in [-L/2, L/2] \times [-L/2, L/2]$   
with periodic boundary conditions

$$u(t, x, -L/2) = u(t, x, L/2)$$

$$u(t, -L/2, y) = u(t, L/2, y)$$

Taking a grid  $\Delta x = \Delta y$  and arranging our unknown vector  $\mathbf{u}$  so that

$$\mathbf{u} = \begin{pmatrix} u_{11} \\ u_{12} \\ \cdot \\ \cdot \\ \cdot \\ u_{1n} \\ u_{21} \\ u_{22} \\ \cdot \\ \cdot \\ \cdot \\ u_{n(n-1)} \\ u_{nn} \end{pmatrix}$$

As in the 1D case we can write the spatially discretized original PDE as an ODE system:

$$\frac{d\mathbf{u}}{dt} = \frac{\kappa}{\Delta x^2} \mathbf{B}\mathbf{u}$$

As in the 1D case,  $\mathbf{B}$  is constructed by taking a second order finite difference approximation for the second derivative of  $u(t, x, y)$ :

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(t, x + \Delta x, y) - 2u(t, x, y) + u(t, x - \Delta x, y)}{\Delta x^2} + \frac{u(t, x, y + \Delta y) - 2u(t, x, y) + u(t, x, y - \Delta y)}{\Delta y^2}$$

In MATLAB

```
% define a function rhs in a file rhs2D.m  
function rhs2D=rhs2D(tspan,u,dummy,k,dx,A)  
rhs2D=(k/dx^2)*A*u;
```

```
% spatial domain of x and y  
Lx=20; Ly=20;
```

```
% discretization points in x and y  
nx=100; ny=nx;
```

```
% build a vector of ones
e1=ones(nx,1);

% construct spatial discretization matrix A
A=spdiags([e1 -2*e1 e1],[-1 0 1],nx,ny);

% periodic boundaries:
A(1,ny)=1; A(nx,1)=1;

% identity matrix of size nx^2
I=eye(nx);

% 2D differentiation matrix using A from 1D
B=kron(I,A)+kron(A,I);

% account for periodicity
x2=linspace(-Lx/2,Lx/2,nx+1);
x=x2(1:nx);

% account for periodicity
y2=linspace(-Ly/2,Ly/2,ny+1);
```

```
y=y2(1:ny);
dx = x(2)-x(1); % or: dx = 1/n

% set up for 2D initial conditions
[X,Y]=meshgrid(x,y);

% generate a Gaussian initial condition matrix
U=exp(-X.^2-Y.^2);

% elements in reshaped initial condition
N=nx*ny;

% reshape into a vector
u=reshape(U,N,1);

% solve the ode for kappa = 1
k = 1;
tspan = [0 1];
[t,y]=ode45('rhs2D',tspan,u,[],k,dx,B);

% plot the solution for the last time
fin_sol = y(end,:);
u_final = reshape(fin_sol,nx,nx)
surf(X,Y,u_final);
```