

# Lecture 23: Time-stepping schemes II.

The diffusion (heat) equation with  $\{\kappa = 1\}$ .

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

Applying the **Method of Lines (MOL)** with a **Forward Euler** time-stepping scheme

$$\begin{aligned} u_n^{(m+1)} &= u_n^{(m)} + \frac{\Delta t}{\Delta x^2} \left( u_{n+1}^{(m)} - 2u_n^{(m)} + u_{n-1}^{(m)} \right) \\ &= u_n^{(m)} + \lambda \left( u_{n+1}^{(m)} - 2u_n^{(m)} + u_{n-1}^{(m)} \right) \end{aligned}$$

Courant, Friedrichs, and Levy (CFL) condition

$$\rightarrow \lambda = \frac{\Delta t}{\Delta x^2}$$

The Forward Euler time-stepping scheme for the heat equation is stable for  $\lambda \leq 1/2$ .

Using second order central differencing in time as well as space results in the **leap-frog (2,2)**

$$\begin{aligned}u_n^{(m+1)} &= u_n^{(m-1)} + \frac{2\Delta t}{\Delta x^2} \left( u_{n+1}^{(m)} - 2u_n^{(m)} + u_{n-1}^{(m)} \right) \\ &= u_n^{(m-1)} + 2\lambda \left( u_{n+1}^{(m)} - 2u_n^{(m)} + u_{n-1}^{(m)} \right)\end{aligned}$$

The leap-frog(2,2) method applied to the heat equation is unstable.

**Implicit scheme: Backward Euler method,**  
is stable for all  $\lambda$ !

$$\begin{aligned}u_n^{(m+1)} &= u_n^{(m)} + \frac{\Delta t}{\Delta x^2} \left( u_{n+1}^{(m+1)} - 2u_n^{(m+1)} + u_{n-1}^{(m+1)} \right) \\ &= u_n^{(m)} + \lambda \left( u_{n+1}^{(m+1)} - 2u_n^{(m+1)} + u_{n-1}^{(m+1)} \right)\end{aligned}$$

Rearranging so that all  $(\cdot)^{(m+1)}$  terms are on the left hand side of the "=", and all  $(\cdot)^{(m)}$  terms on the right hand side.

$$-\lambda u_{n-1}^{(m+1)} + (1 + 2\lambda)u_n^{(m+1)} - \lambda u_{n+1}^{(m+1)} = u_n^{(m)}$$

Which can be written as a matrix system (with a tridiagonal matrix  $\mathbf{A}$ )

$$\mathbf{A}\mathbf{u}^{(m+1)} = \mathbf{u}^{(m)}$$

In MATLAB (Forward Euler and Leap-Frog(2,2)) solve the heat equation with  $\kappa = 1$ , for  $x \in [-L/2, L/2]$  and periodic BCs.

```
% initialize grid size, CFL number and time
L=20;
n=200;
x2=linspace(-L/2,L/2,n+1);  x=x2(1:n);
dx=x(2)-x(1);

dt=0.2;
CFL=dt/(dx^2); % Note different CFL for heat eqn.

Time=4;
time_steps=Time/dt;
t=0:dt:Time;

% initial conditions (gaussian)
u0=exp(-x.^2);
usol(:,1)=u0;
u1=exp(-(x+dt).^2);
usol(:,2)=u1;
```

```

% sparse matrix for second derivative term
e1=ones(n,1);
A=spdiags([e1 -2*e1 e1],[-1 0 1],n,n);

% periodic boundary conditions
A(1,n)=1; A(n,1)=1;

% leap frog (2,2) or euler iteration scheme
for j=1:time_steps-1

% forward euler (start at u1)
u2 = u1 + CFL*A*u1;
u1 = u2;

% leap-frog (2,2) (needs u0 and u1)
% u2 = u0 + 2*CFL*A*u1; % leap frog (2,2)
% u0 = u1; u1 = u2; % leap frog (2,2)

% save solution at time slice in matrix
usol(:,j+2)=u2;
end

```

```

% plot the data
waterfall(x,t,usol');
map=[0 0 0];
colormap(map);

% set x and y limits and fontsize
set(gca,'Xlim',[-L/2 L/2],'Xtick',[-L/2 0 L/2], ...
'FontSize',[20]);
set(gca,'Ylim',[0 Time],'Ytick',[0 Time/2 Time], ..
'FontSize',[20]);
view(25,40)

% set axis labels and fonts
xl=xlabel('x');
yl=ylabel('t');
zl=zlabel('u');
set(xl,'FontSize',[20]);
set(yl,'FontSize',[20]);
set(zl,'FontSize',[20]);

```

It is worthwhile to note that generally the Euler methods work better for problems of diffusive nature and leap-frog schemes work better for wave propagation problems.