Brunel University London

ME2610
Engineering Mathematics and Programming

12th November 2020

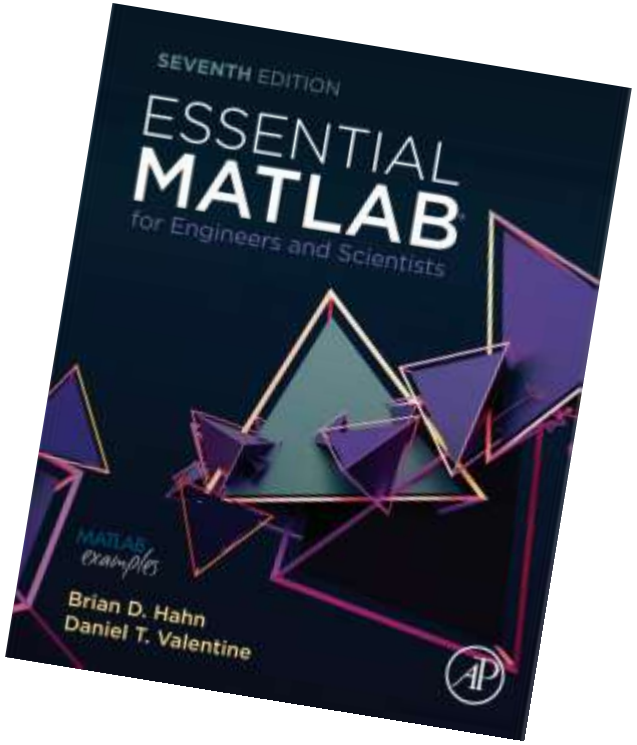
Dr Edward Smith

Room 105

Howell Building

Summary

- Brief Recap
- Partial Differential Equation - Temporal spatial
  - Initial condition
  - Boundary conditions
  - Convective terms and forcing
- Partial Differential Equation - Two dimensional Laplace equations
  - Reminder of 2D fields and partial derivatives
  - Solution of Laplace's equation



Essential Matlab (EM)
http://tinyurl.com/yy53shga

This session will be recorded

Learning Aims

- **LO2:** Understanding how to employ programming to solve basic engineering computational problems.
- **LO4:** Applying best-practice programming techniques to solve Mathematical models of Engineering problems.
- **LO5:** Understanding the usefulness of programming techniques in the process of solving Engineering problems.
- **LO6:** Presenting computational results in a clear and concise manner including validation and verification.

Registration and Questions

https://brunel.onlinesurveys.ac.uk/feedback_me2610

- Use the QR code to go to feedback
- You can ask questions or make comments at any time, either linked to your name (if you put it in) or anonymously (if you don't)

Plan for Course

Week	Lecture No	Count M/	Lecture Content	Tutorial	Deadline	Date
1	1	1	Interpolation methods			29/09/2020
1	2	1	Introduction			29/09/2020
1	3	2	Interpolation methods			01/10/2020
1	4	2	Data types, matrices and arrays	TEST Matlab	|	01/10/2020
2	5	3	Interpolation methods	Basic arrays	|	06/10/2020
2	6	3	For and if statements	Matrices and simulatanous	|	06/10/2020
2	7	4	Root Finding	Interfaces and tests	↓	08/10/2020
2	8	4	Functions and Interfaces	For and if statements	**TEST Matlab**	08/10/2020
					|	
					|	
					|	
					|	15/10/2020
4	9	5	Root Finding	Interpolation	|	20/10/2020
4	10	5	Interpolation Numerics	Interpolation	|	20/10/2020
4	11	6	Root Finding	Root finding	↓	22/10/2020
4	12	6	Root Finding Numerics	Root finding	**Functions**	22/10/2020

Plan for Course

5	13	7	Integration Methods	Trapizum rule	|	27/10/2020
5	14	7	Integration Numerics	Simpson Rule	|	27/10/2020
5	15	8	Integration methods	Gauss integration	↓	29/10/2020
5	16	9	Gauss integration	Gauss integration	**Root/interpolation**	29/10/2020
6	17	10	Diff. equations (integrating factors, order)	Basic Finite difference	|	03/11/2020
6	18	8	Intro Finite Difference	Basic Finite difference	|	03/11/2020
6	19	11	Diff. equations (integrating factors, order)	Basic Finite difference	↓	05/11/2020
6	20	9	Explicit + 2nd order Finite Difference	Basic Finite difference	**Integration**	05/11/2020
7	21	12	Diff. equations (integrating factors, order)	1D ODE	|	10/11/2020
7	22	10	Implict Finite Difference	1D ODE	|	10/11/2020
7	23	13	2D unsteady convection from 1st principles	SIR Equation	↓	12/11/2020
7	24	11	2D Finite Difference	SIR Equation	**1D ODE**	12/11/2020
8	25	14	Vector functions/ Jacobian Newton-Raphson 2D	2D PDE	|	17/11/2020
8	26	12	Validation and Verification	2D PDE	|	17/11/2020
8	27	15	Vector functions/ Jacobian Newton-Raphson 2D	2D PDE	|	19/11/2020
8	28	16	Vector functions/ Jacobian Newton-Raphson 2D	2D PDE	|	19/11/2020
9	29	17	Laplace Transforms	Assignment help	|	24/11/2020
9	30	18	Laplace Transforms	Assignment help	|	24/11/2020
9	31	19	Laplace Transforms	Assignment help	↓	26/11/2020
9	32	20	Laplace Transforms	Assignment help	**Assignment 2D PDE**	26/11/2020

Differential Equations

- Equations which include derivatives are differential equations, e.g.

$$\frac{df}{dx} = 0 \qquad \frac{d^2f}{dx^2} = 0 \qquad \frac{d^2f}{dx^2} + f = 0$$

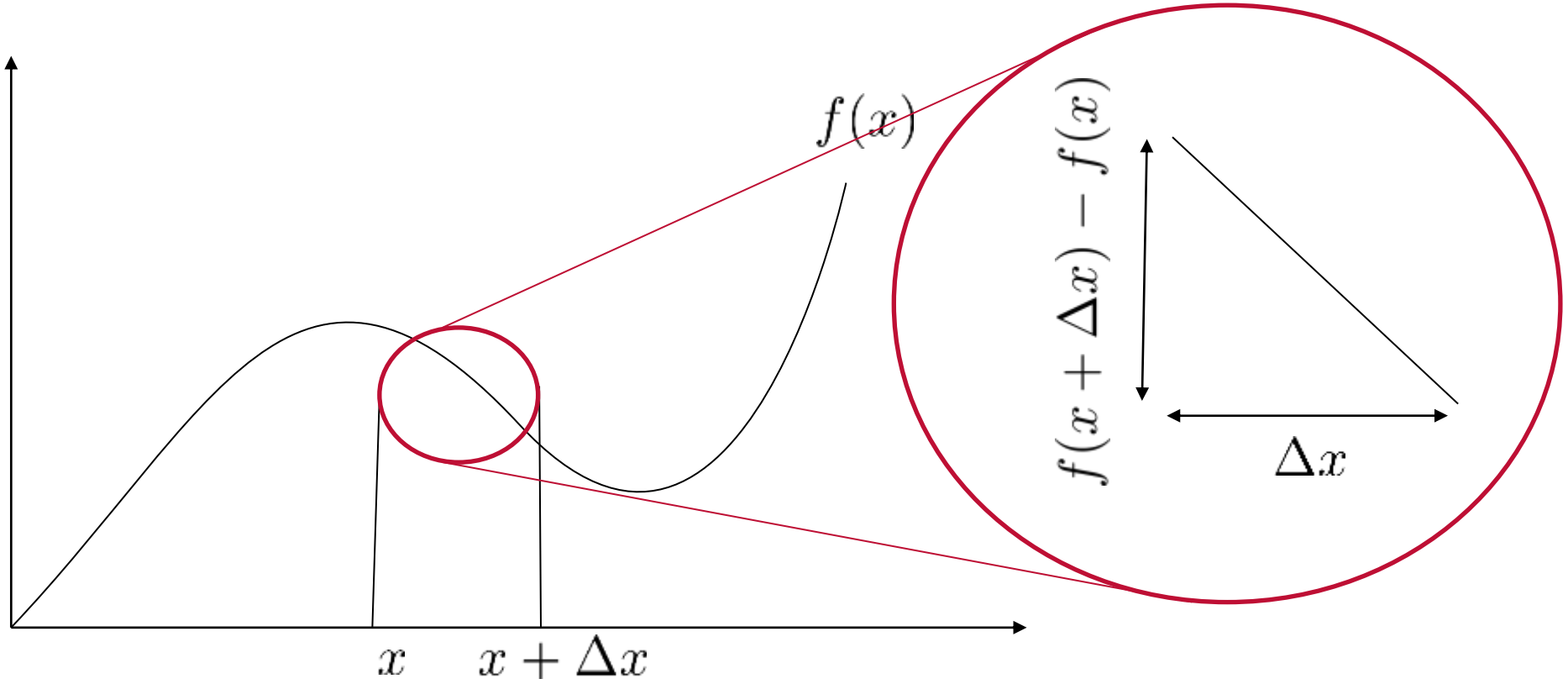
- These are the same as any other equation, for example the equation for a line or Newton's law

$$y = mx + c \qquad F = ma$$

- Which can also be written as differential equations

$$y = \frac{dy}{dx}x + c \qquad F = m\frac{d^2r}{dt^2}$$

Approximating a Derivative



So instead we approximate by not taking the limiting case

$$\frac{df}{dx} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Numerical Solutions to 1st and 2nd Order Terms

- First order derivatives

$$\frac{df}{dx} \approx \frac{f_{i+1} - f_i}{\Delta x}$$

- Second order derivatives

$$\frac{d^2 f}{dx^2} \approx \frac{f_{i+1} - 2f_i + f_{i-1}}{(\Delta x)^2}$$

- We write as code in the same way (rearranged to get i+1 value)

$$\frac{df}{dx} = a$$

```
f(i+1) = f(i) + a*dx
```

$$\frac{d^2 f}{dx^2} = b$$

```
f(i+1) = 2*f(i) - f(i-1) + b*dx^2
```

Second Order - Initial Value Problem (Spring Mass)

- Recall the spring mass system you studied in last years dynamics lab

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

Mass damping spring constant

$$m\frac{x_{i+1} - 2x_i + x_{i-1}}{\Delta t^2} + c\frac{x_i - x_{i-1}}{\Delta t} + kx_i = 0$$

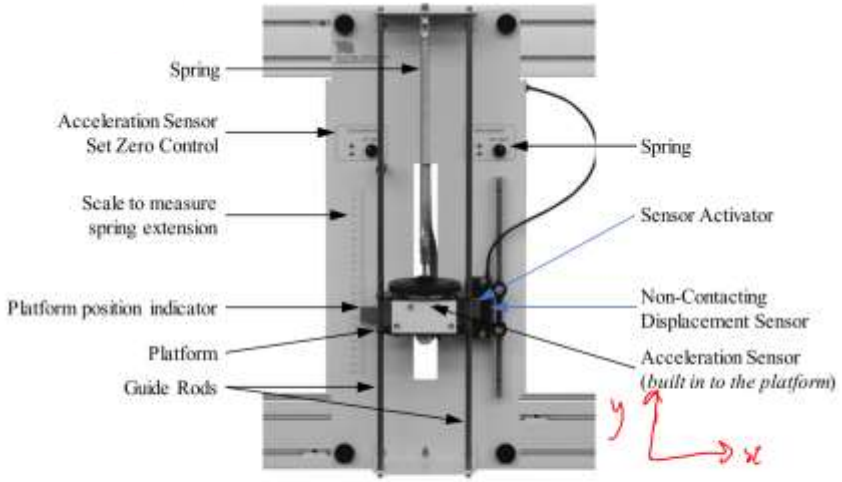


Figure 2 – Free vibration of a mass-spring system shown fitted to the free vibration test frame.

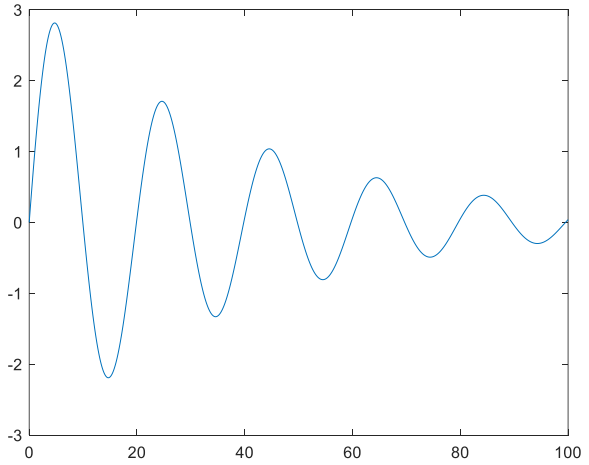
- Put discrete forms in equations and rearrange to get x_{i+1} (using x0 and v0) and defining mass m, spring constant k and damping (rate oscillation decreases) c.

$$x_{i+1} = 2x_i - x_{i-1} - \frac{c}{m}\Delta t(x_i - x_{i-1}) - \Delta t^2\frac{k}{m}x_i$$

Second Order - Initial Value Problem (Spring Mass)

- Put discrete forms in equations and rearrange to get x_{i+1} (using x0 and v0) and defining mass m, spring constant k and damping (rate oscillation decreases) c.

```
m = 1; %Mass
c = 0.05; %Damping
k = 0.1; %Spring constant
x0 = 0; %Initial position
v0 = 1; %Initial velocity
x(1) = x0;
x(2) = v0*dt + x(1);
M = 1000; dt=0.1;
for i =2:M
    x(i+1) = 2*x(i) - x(i-1) - dt*(c/m)*(x(i) - x(i-1)) - dt^2*(k/m)*x(i);
end
plot(0:dt:M*dt,x)
```



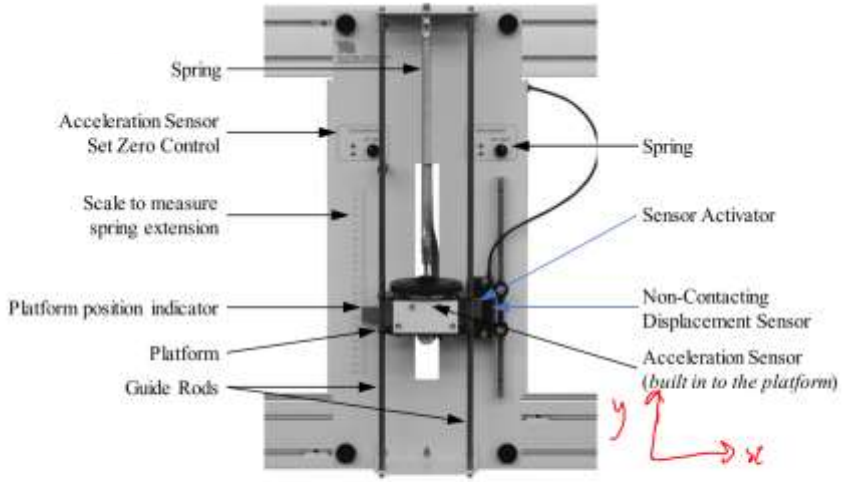
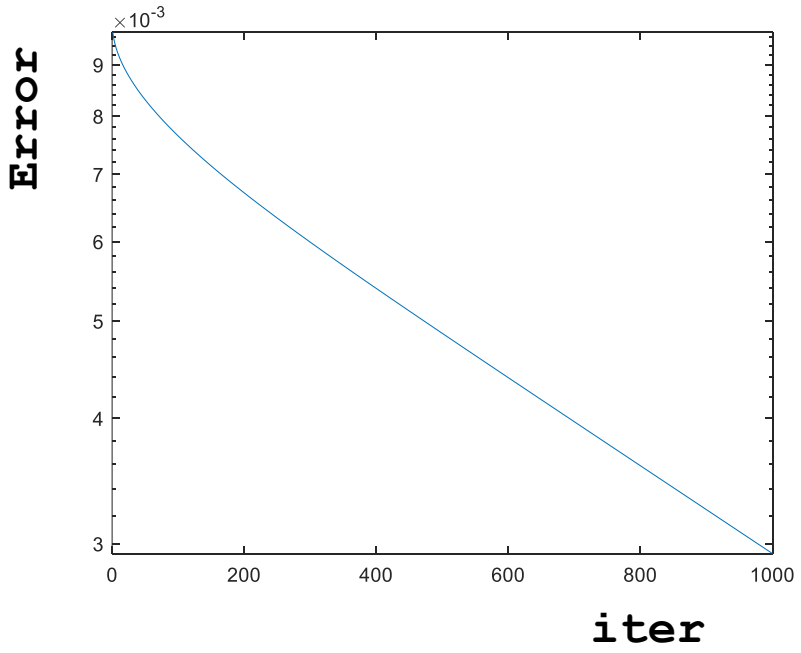


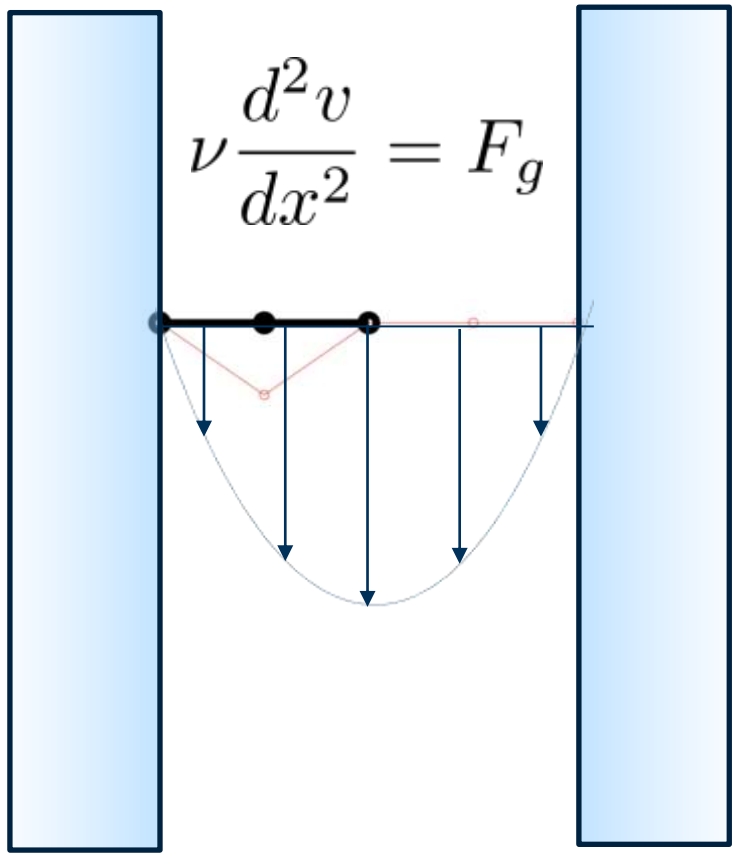
Figure 2 – Free vibration of a mass-spring system shown fitted to the free vibration test frame.

$$x_{i+1} = 2x_i - x_{i-1} - \frac{c}{m}\Delta t(x_i - x_{i-1}) - \Delta t^2 \frac{k}{m} x_i$$

Boundary Value Problem (flow between 2 walls)

- Iterate diffusive flow driven by gravity between two walls to steady state, v=0 at the walls x=0 and x=L with L=1, F_g=1 and nu=0.1
  - 2nd derivative models fluid diffusion with viscosity *v* (Greek nu)

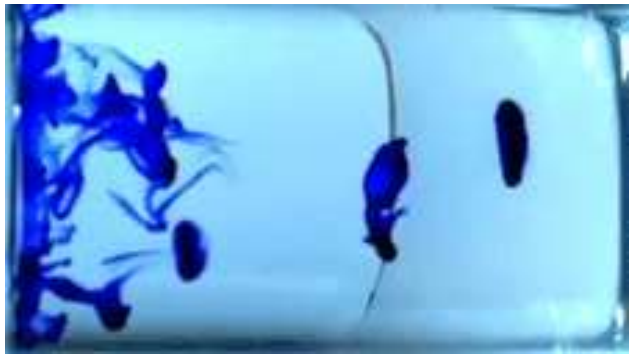




Time Evolving Equations

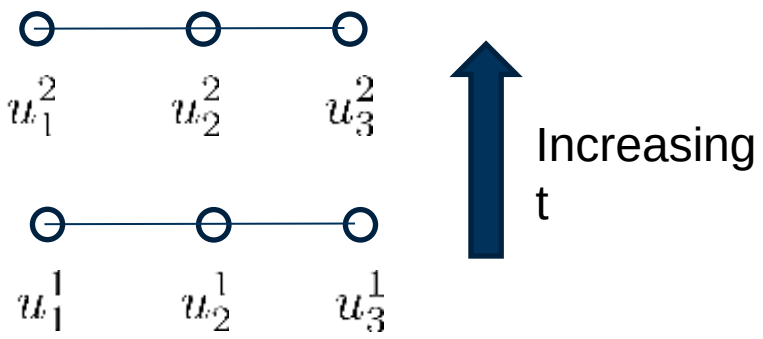
- Partial differential equations can include changes in time and 3 dimensions in space – we will start with 1D and varying in time
  - We have a time evolving term on the left
  - We have a spatial diffusion term on the right

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



- We discretise this using the same formulas we have seen already
  - However, we denote time as a superscript
  - Spatial components are subscripts as previously

$$\frac{u_i^{t+1} - u_i^t}{\Delta t} = \nu \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2}$$



Time Evolving Equations

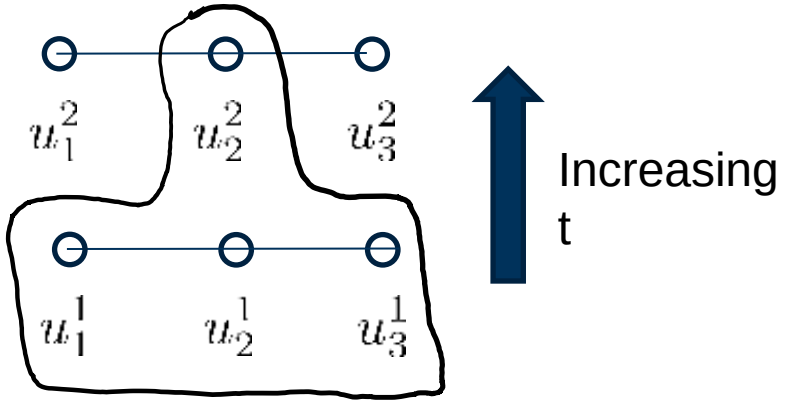
- We discretise this using the same formulas we have seen already
  - However, we denote time as a superscript
  - Spatial components are subscripts as previously

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(end/2) = 1; utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

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

$$\frac{u_i^{t+1} - u_i^t}{\Delta t} = \nu \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2}$$

$$u_i^{t+1} = u_i^t + \nu \Delta t \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2}$$

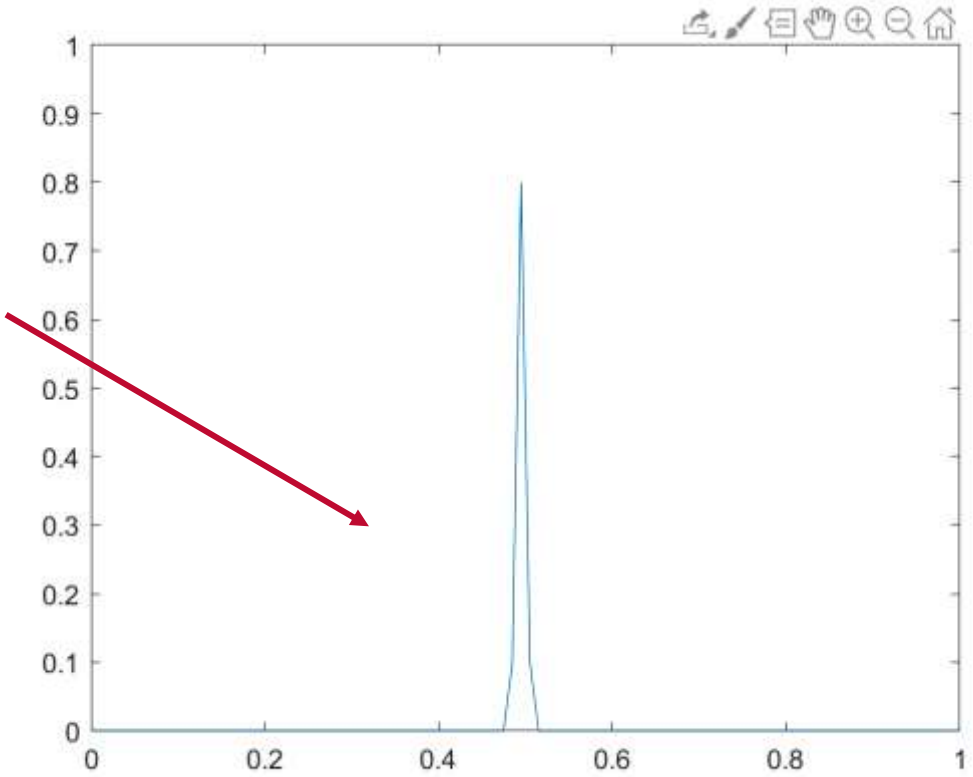


Time Evolving Equations – initial conditions

- This models time evolving diffusion and the explicit iteration can now be though of as evolving the system in time – **initial values matter**

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(end/2) = 1; utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

Initial value of 1 in the middle

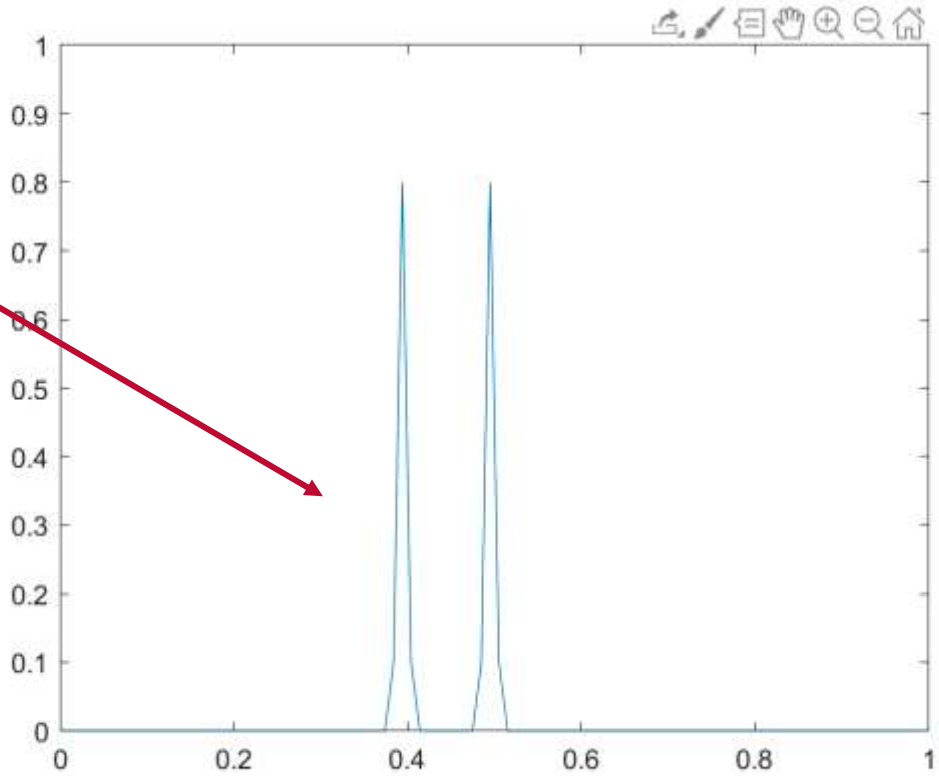


Time Evolving Equations – initial conditions

- This models time evolving diffusion – the explicit iteration can now be though of as evolving the system in time – **initial values matter**

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(4*end/10)=1; u(5*end/10)=1; utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

2 values of 1 in the middle

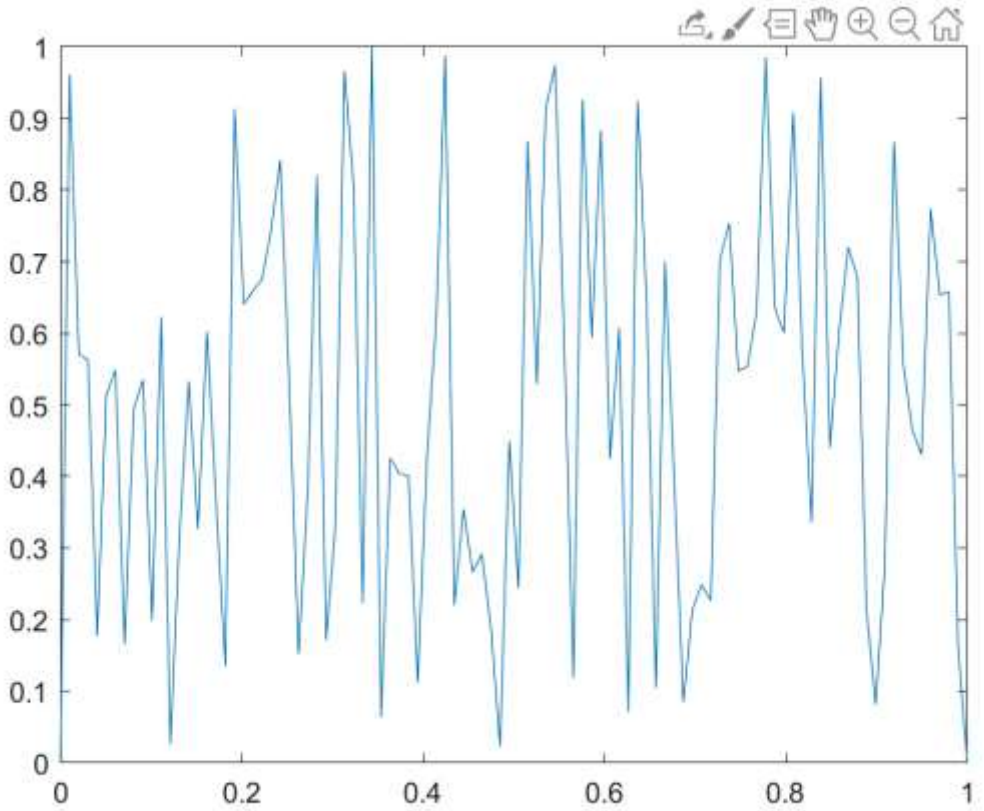


Time Evolving Equations – initial conditions

- This models time evolving diffusion – the explicit iteration can now be though of as evolving the system in time – **initial values matter**

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(:) = rand(100,1); utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

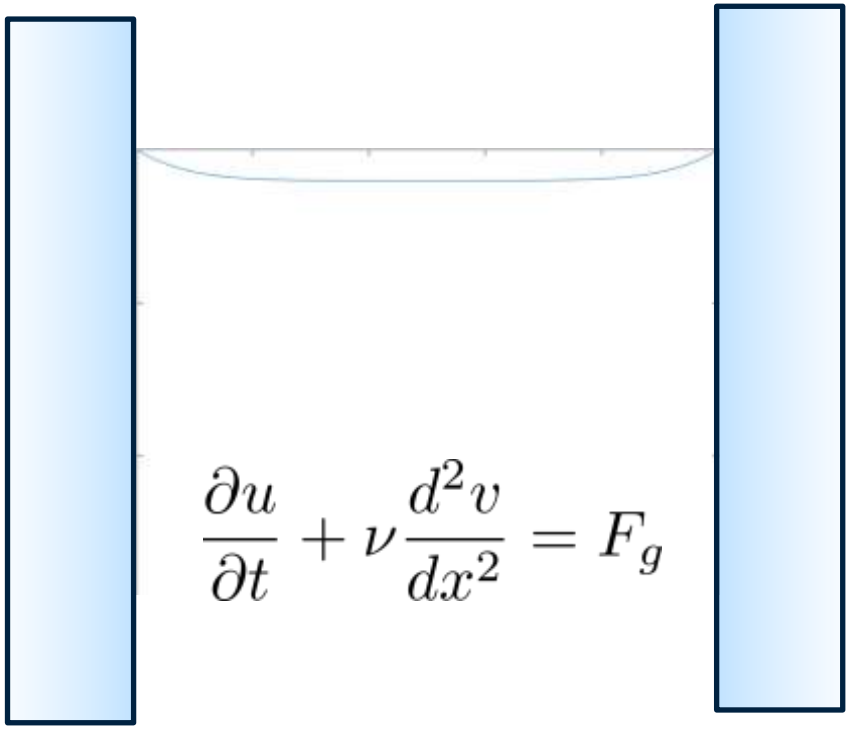
Random noise



Time Evolving Equations – Applied Force

- Applying the force term used previously, we also get the same parabolic profile – this is the velocity evolving in time

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001; Fg=1;
x = linspace(0,L,M); u = zeros(M,1);
utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2 - dt*Fg;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```



$$\frac{\partial u}{\partial t} + \nu \frac{d^2 v}{dx^2} = F_g$$

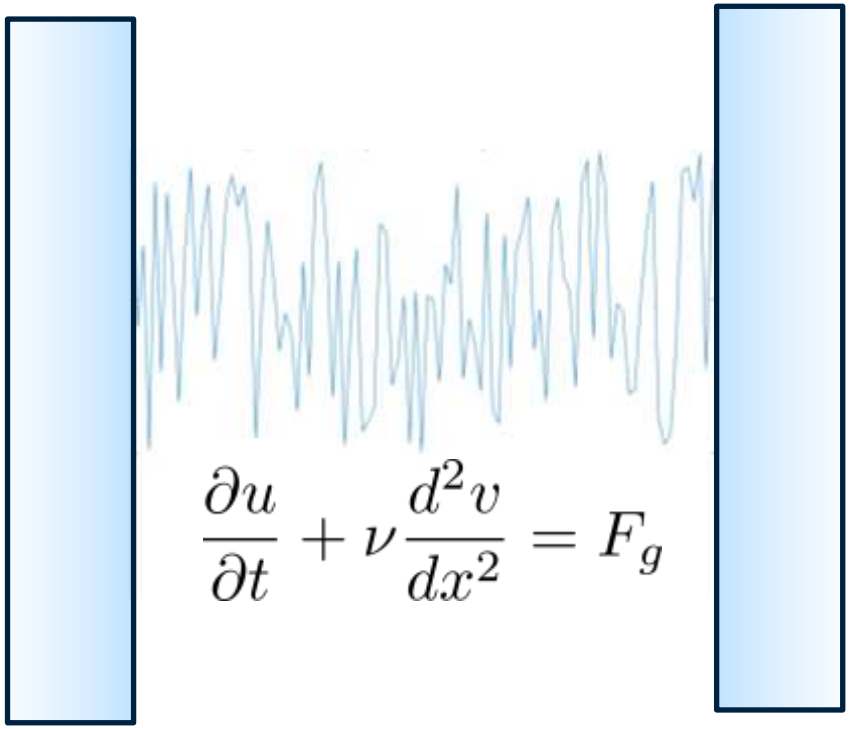
$$u_i^{t+1} = u_i^t + \nu \Delta t \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2} + \Delta t F_g$$

Time Evolving Equations – Applied Force

- Applying the force term used previously, we also get the same parabolic profile (even when starting from different initial values)

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001; Fg=1;
x = linspace(0,L,M); u = zeros(M,1);
u(:) = rand(100,1); utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 0; u(M) = 0;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2 - dt*Fg;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

Random noise



$$\frac{\partial u}{\partial t} + \nu \frac{d^2 v}{dx^2} = F_g$$

$$u_i^{t+1} = u_i^t + \nu \Delta t \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2} + \Delta t F_g$$

Boundary Condition

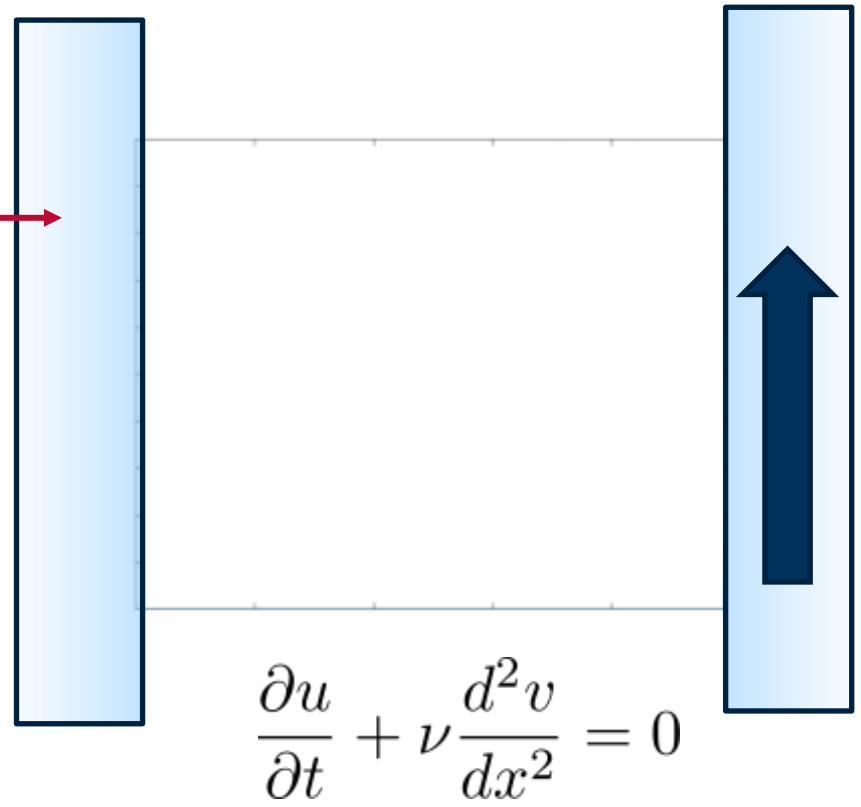
- The values at the edge are the boundary conditions
- We have previously set to zero to model the no-slip condition in a fluid
- We could set to non-zero values to model a moving wall in a fluid
- We can also choose more exotic boundary conditions

Moving Wall Boundary Conditions

- A channel with a moving wall (e.g. inside a bearing, the gap between an engine piston head and wall or a fluid film)

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = 1; u(M) = 1;
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

Wall velocity set to 1



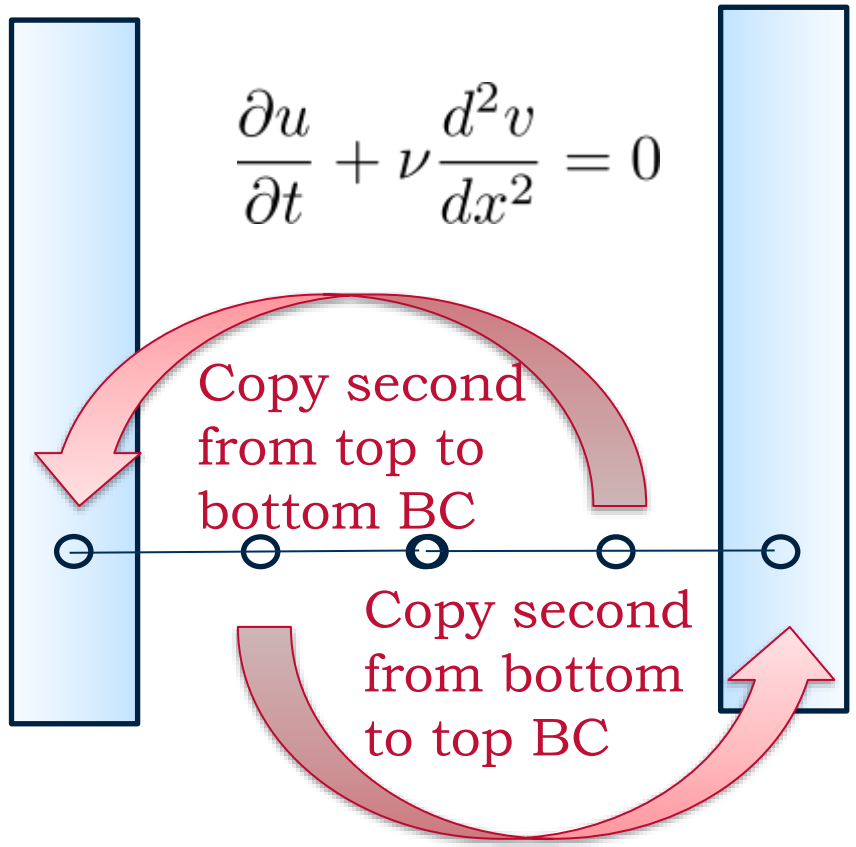
$$\frac{\partial u}{\partial t} + \nu \frac{d^2 v}{dx^2} = 0$$

Periodic Boundary Conditions (BC)

- A channel with a moving wall (e.g. inside a bearing, the gap between an engine piston head and wall or a fluid film)

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(2) = 1; utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = u(M-1); u(M) = u(2);
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

Copy one side to other

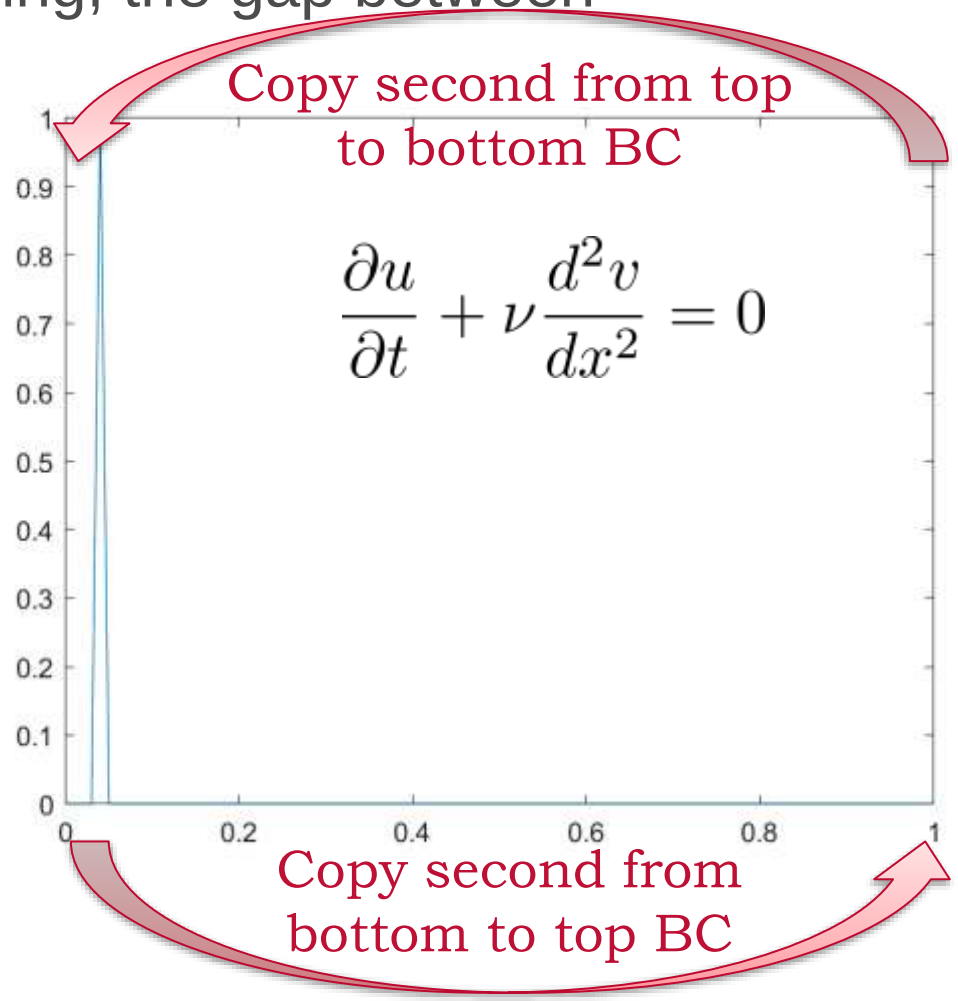


Periodic Boundary Conditions (BC)

- A channel with a moving wall (e.g. inside a bearing, the gap between an engine piston head and wall or a fluid film)

```
%Define coefficients
nu=0.1; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(5) = 1; utp1 = u;
%Iterate in time
for t=1:1000
    u(1) = u(M-1); u(M) = u(2);
    for i=2:M-1
        utp1(i)=u(i)+dt*nu*(u(i+1) ...
                -2*u(i)+u(i-1))/dx^2;
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

Initial value of 1 near the edge



Burgers Equation

- The same equation as before but we add in a convection term
  - Travelling Wave Solution

```
%Define coefficients
nu=0.01; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(1:end/2) = 1; utp1 = u;
%Iterate in time
for t=1:10000
    %Periodic
    u(1) = u(M-1); u(M) = u(2);
    for i=2:M-1
        utp1(i)=u(i)+dt*( u(i)*(u(i+1)-u(i))/dx ...
                          +nu*(u(i+1)-2*u(i)+u(i-1))/dx^2);
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \nu\frac{d^2v}{dx^2} = 0$$

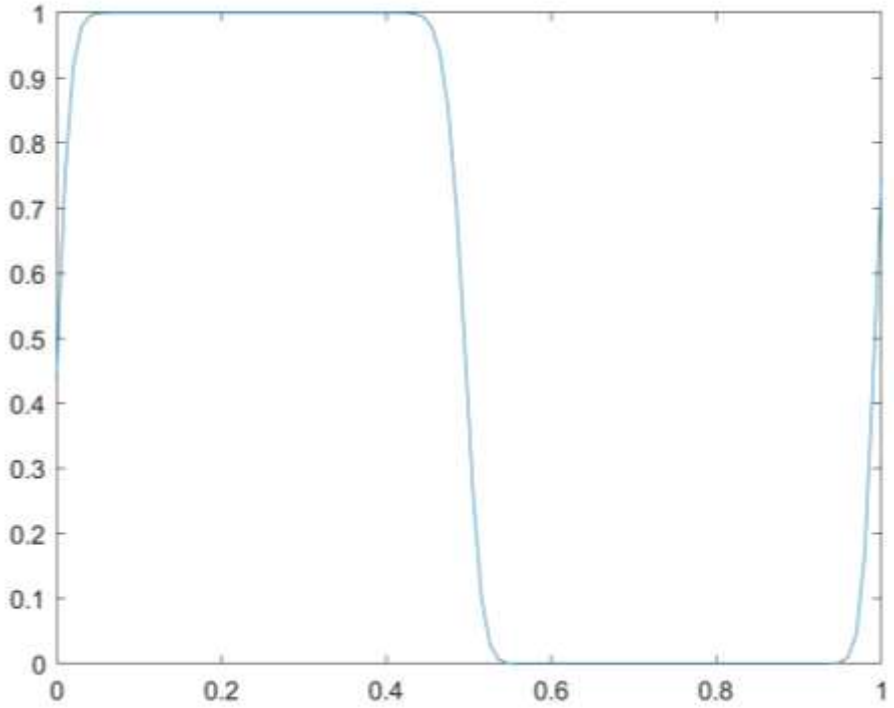
$$\frac{u_i^{t+1} - u_i^t}{\Delta t} = u_i^t\frac{u_{i+1}^t - u_i^t}{\Delta x} + \nu\frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2}$$

Note dt multiplies everything on right hand side

Burgers Equation

- The same equation as before but we add in a convection term
  - Travelling Wave Solution convected over periodic boundary while diffusing

```
%Define coefficients
nu=0.01; L=1; M = 100; dx = L/M; dt=0.0001;
x = linspace(0,L,M); u = zeros(M,1);
u(1:end/2) = 1; utp1 = u;
%Iterate in time
for t=1:10000
    %Periodic
    u(1) = u(M-1); u(M) = u(2);
    for i=2:M-1
        utp1(i)=u(i)+dt*( u(i)*(u(i+1)-u(i))/dx ...
              +nu*(u(i+1)-2*u(i)+u(i-1))/dx^2);
    end
    plot(x,utp1); pause(0.1)
    u = utp1;
end
```



Recap

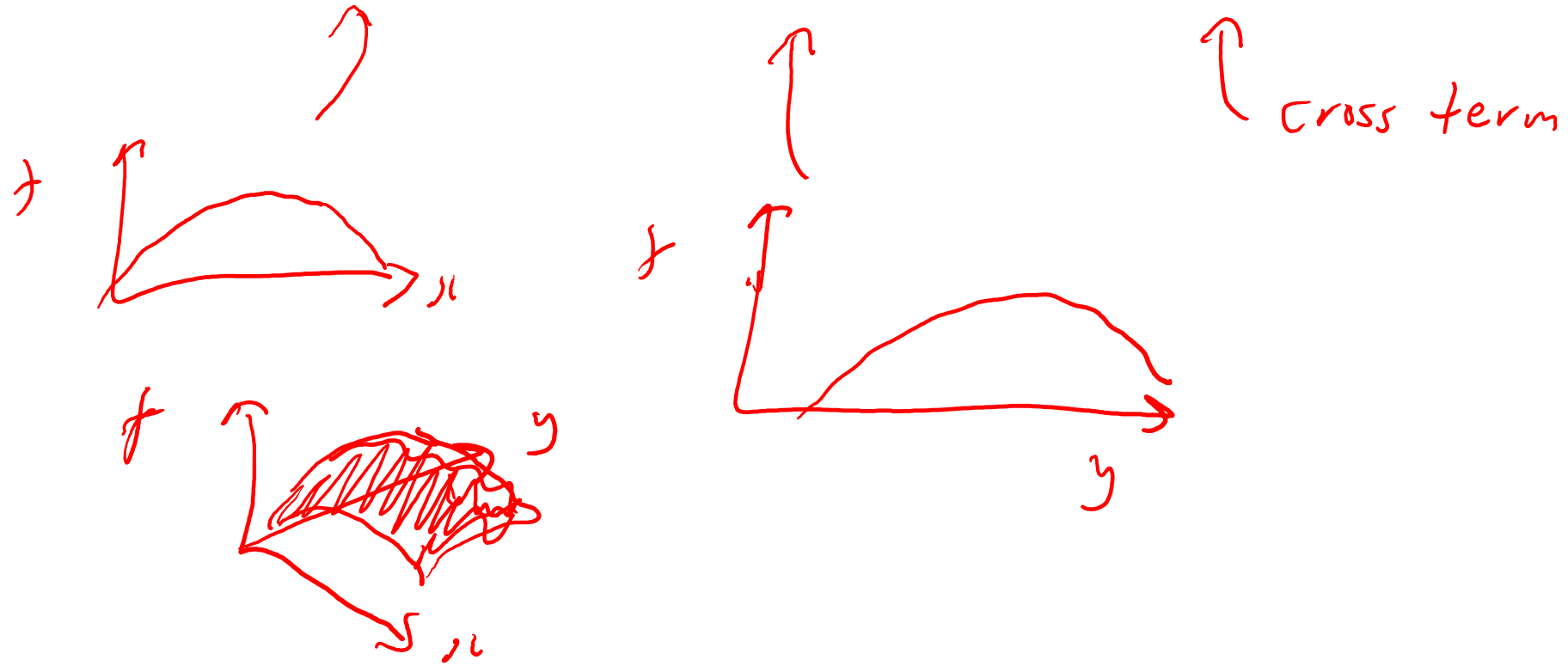
- This may seem like a lot, but we use the same code with small tweaks
  - Setting the initial field to zero, random numbers or some point value
  - Setting boundary values to zero (no slip) or a number (moving or heated wall, etc)
  - Using periodic boundaries so what goes out comes back in
  - Applying a force term to the time evolving solution so it is driven towards a particular solution (e.g. gravity accelerated)
  - Adding in a non-linear convective term

$$u_i^{t+1} = u_i^t + \Delta t \left[\underbrace{u_i^t \frac{u_{i+1}^t - u_i^t}{\Delta x}}_{\text{Convection}} + \underbrace{\nu \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{\Delta x^2}}_{\text{Diffusion}} + \underbrace{F_g}_{\text{Force}} \right]$$

Two Dimensional Field

- Consider the example field described by an x-y polynomial

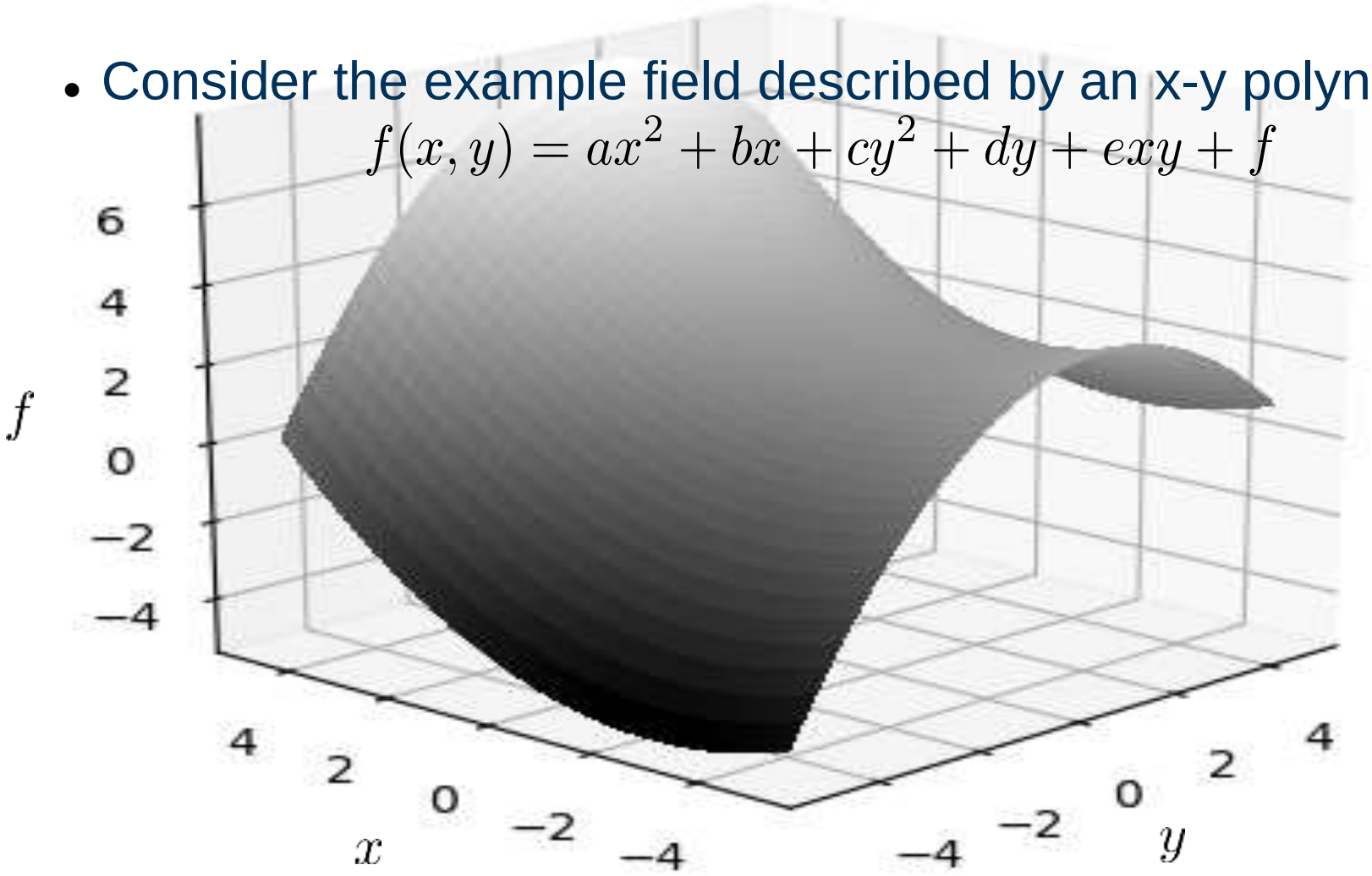
$$f(x,y) = ax^2 + bx + cy^2 + dy + exy + f$$



Two Dimensional Field

- Consider the example field described by an x-y polynomial

$$f(x,y) = ax^2 + bx + cy^2 + dy + exy + f$$



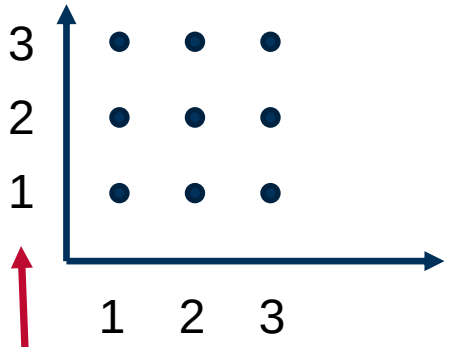
- A 2D field is a function of two variables

$$f = f(x,y)$$

- Show here in 3D for visualisation
- Assumed to be a continuous function

Plotting a 2D Field in Excel

- Plotted in Excel – create a grid of x and y values then plot f(x,y)

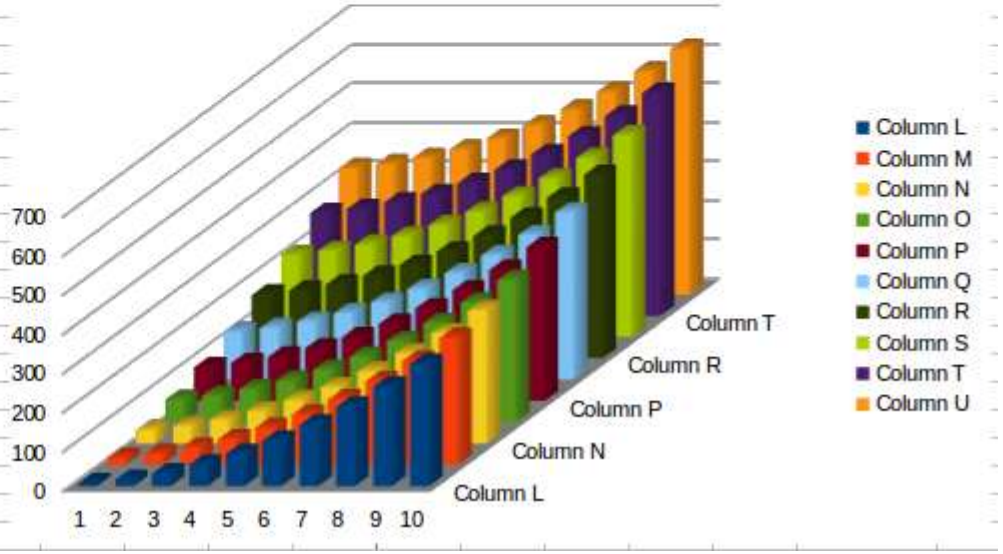


X values at each point in a grid

Corresponding Y values at each point in a grid

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	x											f										
2	1	2	3	4	5	6	7	8	9	10		7	18	35	58	87	122	163	210	263	322	
3	1	2	3	4	5	6	7	8	9	10		17	28	45	68	97	132	173	220	273	332	
4	1	2	3	4	5	6	7	8	9	10		33	44	61	84	113	148	189	236	289	348	
5	1	2	3	4	5	6	7	8	9	10		55	66	83	106	135	170	211	258	311	370	
6	1	2	3	4	5	6	7	8	9	10		83	94	111	134	163	198	239	286	339	398	
7	1	2	3	4	5	6	7	8	9	10		117	128	145	168	197	232	273	320	373	432	
8	1	2	3	4	5	6	7	8	9	10		157	168	185	208	237	272	313	360	413	472	
9	1	2	3	4	5	6	7	8	9	10		203	214	231	254	283	318	359	406	459	518	
10	1	2	3	4	5	6	7	8	9	10		255	266	283	306	335	370	411	458	511	570	
11	1	2	3	4	5	6	7	8	9	10		313	324	341	364	393	428	469	516	569	628	
12																						
13	y																					
14	1	1	1	1	1	1	1	1	1	1												
15	2	2	2	2	2	2	2	2	2	2												
16	3	3	3	3	3	3	3	3	3	3												
17	4	4	4	4	4	4	4	4	4	4												
18	5	5	5	5	5	5	5	5	5	5												
19	6	6	6	6	6	6	6	6	6	6												
20	7	7	7	7	7	7	7	7	7	7												
21	8	8	8	8	8	8	8	8	8	8												
22	9	9	9	9	9	9	9	9	9	9												
23	10	10	10	10	10	10	10	10	10	10												
24																						
25		coefficients																				
26		a	3																			
27		b	2																			
28		c	3																			
29		d	1																			
30		e	0																			
31		f	-2																			
32																						
33																						
34																						

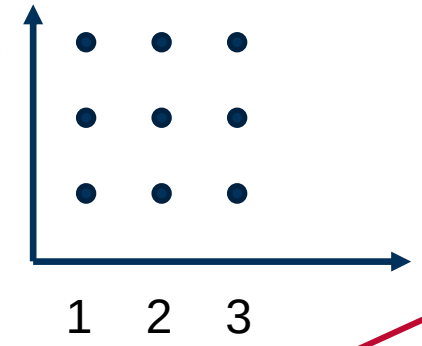
$$f(x,y) = ax^2 + bx + cy^2 + dy + exy + f$$

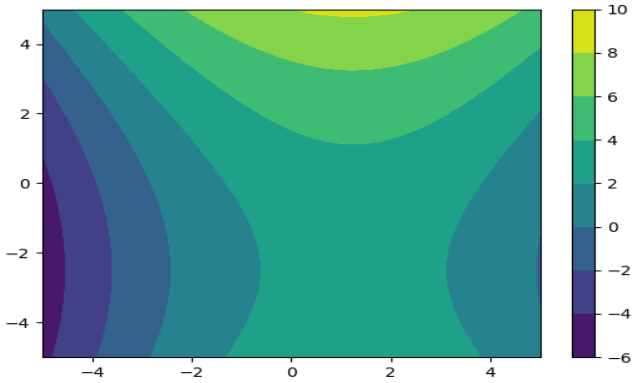


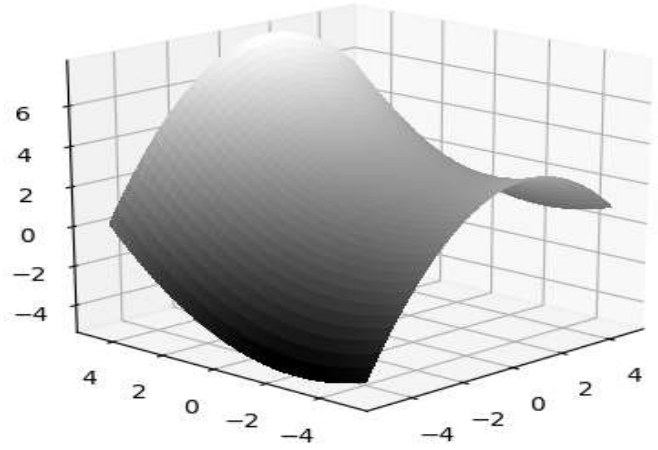
Plotting a 2D Field in MATLAB

- Plotted in MATLAB – using meshgrid for x and y values then plot f(x,y)

```
x = linspace(-5, 5., 100);
y = linspace(-5, 5., 100);
[X, Y] = meshgrid(x, y);

a = -0.2; b = 0.5; c=0.1;
d=0.5; e=0.; f=3.

f = a*X.^2 + b*X + c*Y.^2 + d*Y + e*X.*Y + f;

contourf(X, Y, f)
colorbar

surf(X, Y, f) %To get 3D like plot seen previously
```



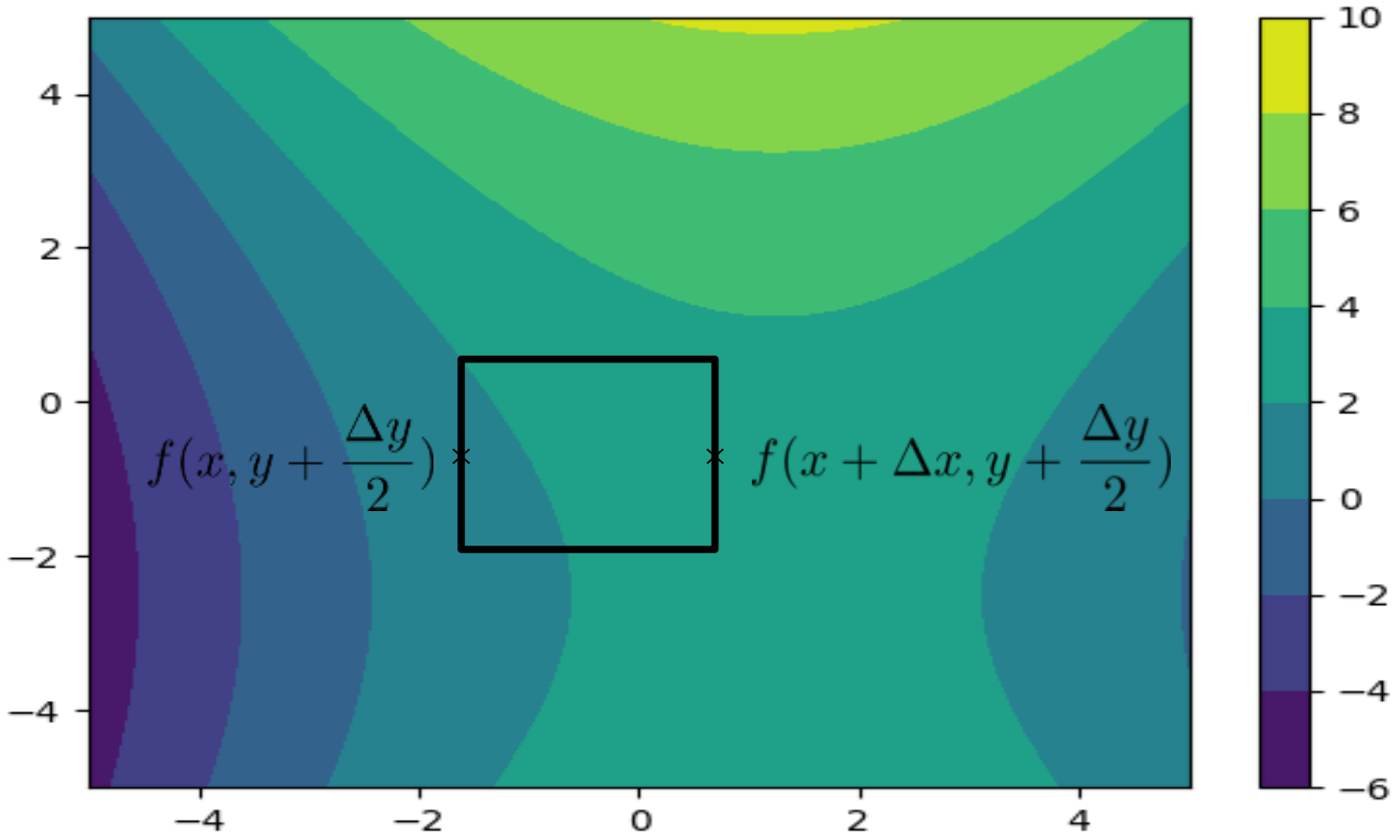




Defining an element of a 2D Field

- Contour plot

$$f(x,y) = ax^2 + bx + cy^2 + dy + exy + f$$



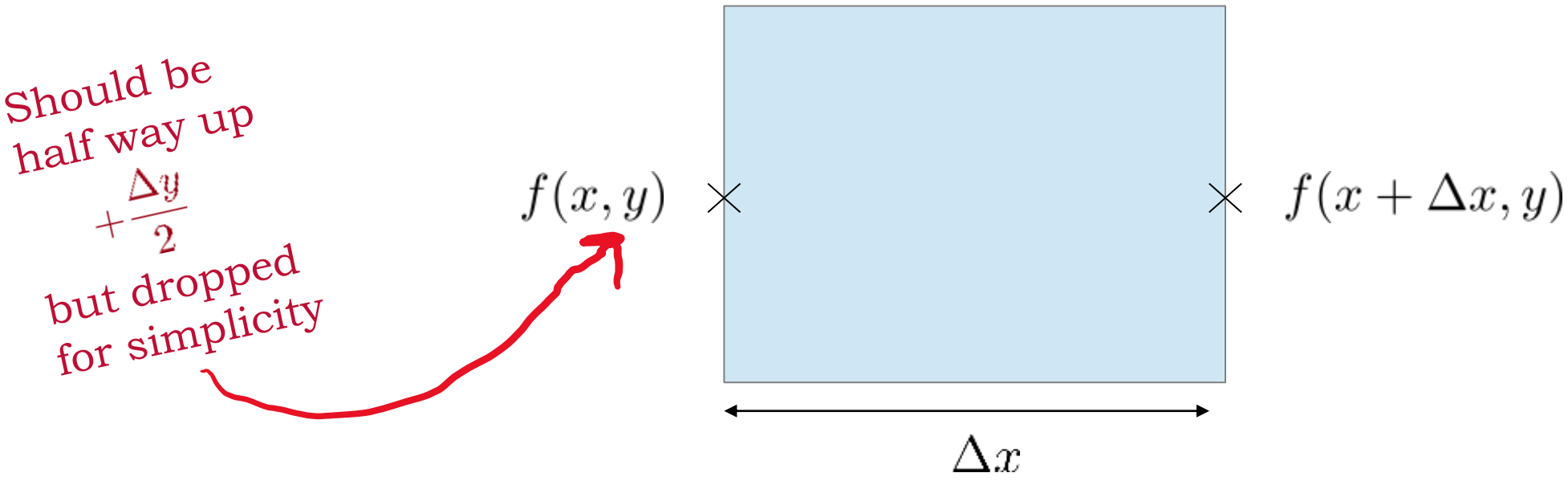
- Limit is a continuous function
- Here a function of two variables

$$f = f(x,y)$$

- As we move in either x or y direction the value of f changes

Two Dimensions and Partial Derivatives

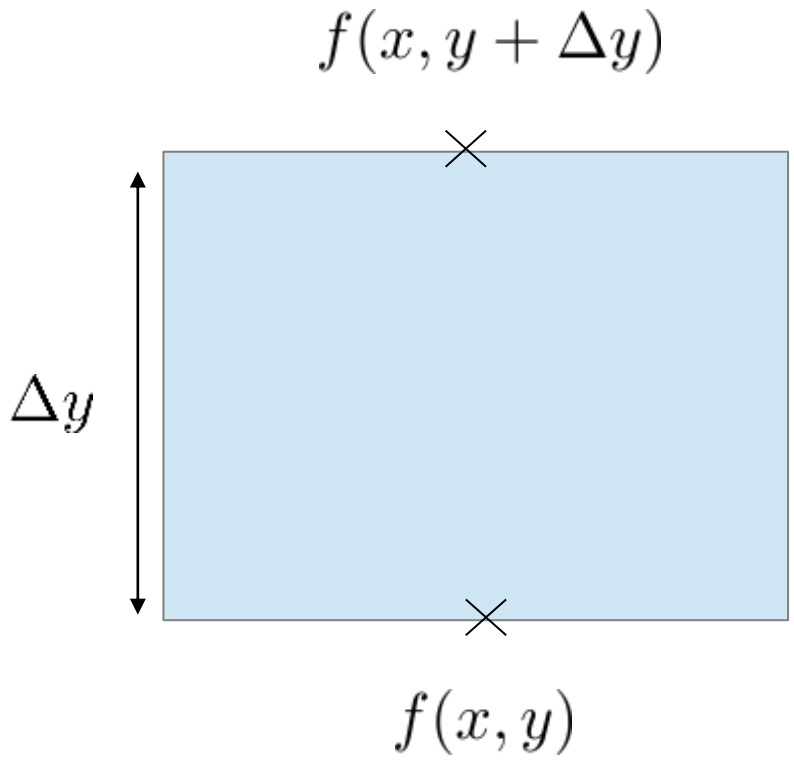
- Change in x keeping y constant (taken half way up element)



- Note we have dropped the $\Delta y/2$ terms for simplicity

$$\left.\frac{\partial f}{\partial x}\right|_{y \text{ constant}} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

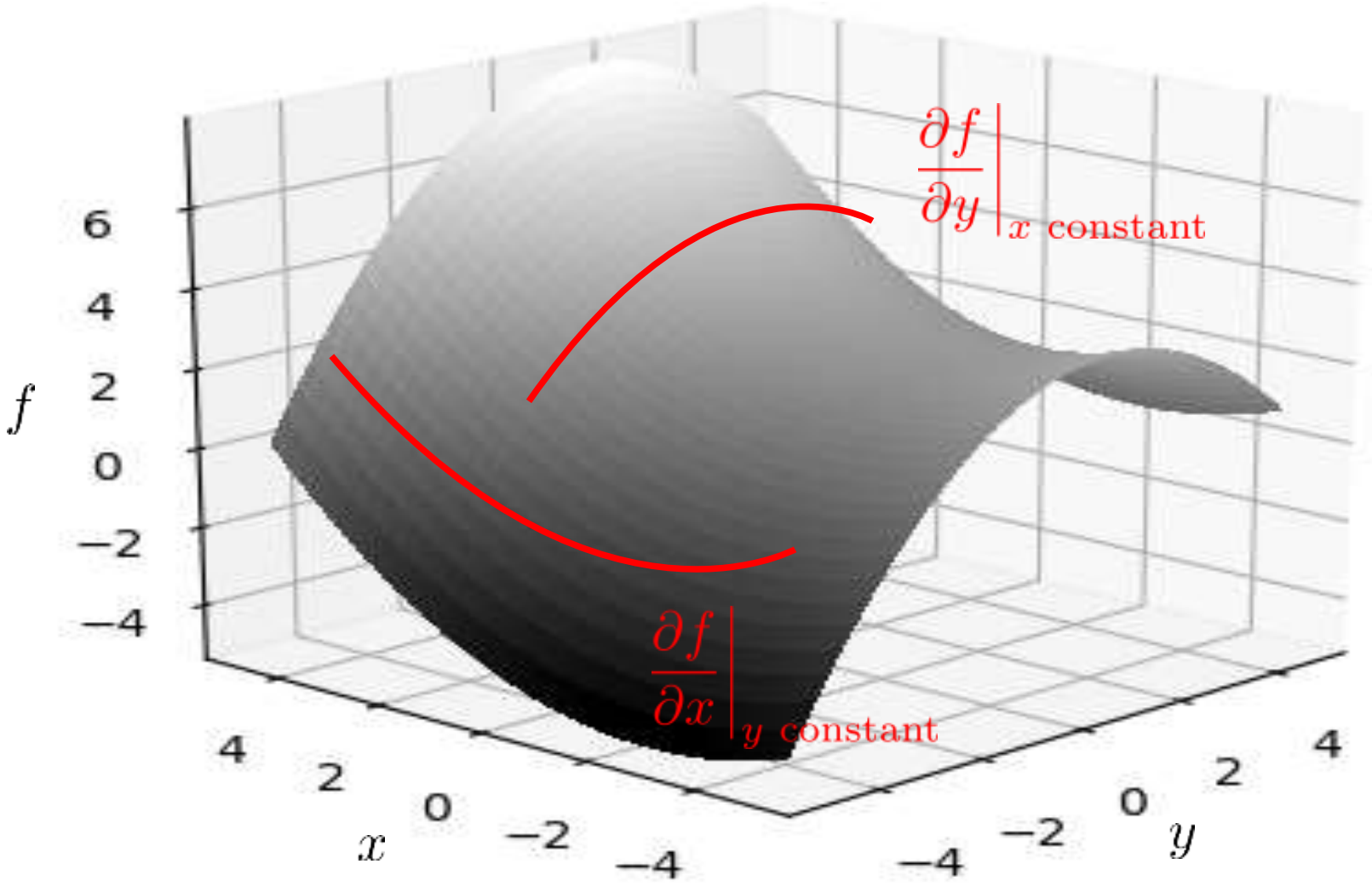
Two Dimensions and Partial Derivatives



- Note we have dropped the $\Delta x/2$ terms for simplicity

$$\left.\frac{\partial f}{\partial y}\right|_{x \text{ constant}} = \lim_{\Delta y \to 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y}$$

Two Dimensions and Partial Derivatives



Consider a function of two variables

$$f = f(x, y)$$

Change in x $\left.\frac{\partial f}{\partial x}\right|_{y \text{ constant}}$

Change in y $\left.\frac{\partial f}{\partial y}\right|_{x \text{ constant}}$

$$f(x, y) = ax^2 + bx + cy^2 + dy + exy + f$$

Two Dimensions and Partial Derivatives

- Consider the example field described by an x-y polynomial

$$f(x,y) = ax^2 + bx + cy^2 + dy + exy + f$$

- We can calculate the derivatives at any point

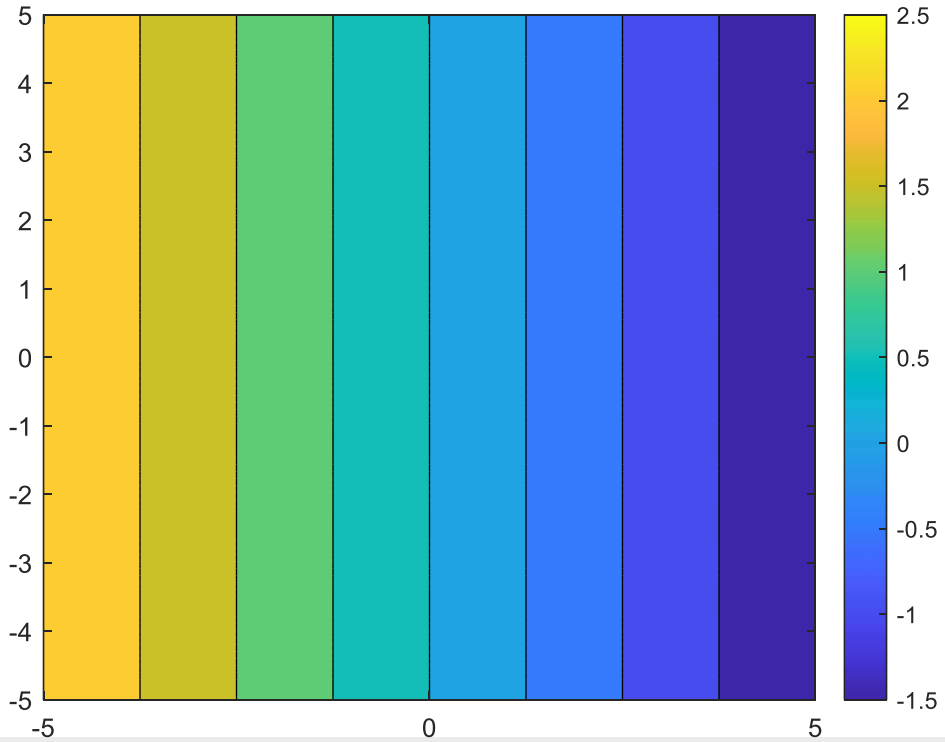
$$\frac{\partial f}{\partial x}|_{y \text{ constant}} = 2ax + b + ey \qquad \frac{\partial f^2}{\partial x^2}|_{y \text{ constant}} = 2a$$

$$\frac{\partial f}{\partial y}|_{x \text{ constant}} = 2cy + d + ex \qquad \frac{\partial f^2}{\partial y^2}|_{y \text{ constant}} = 2c$$

Plotting the Derivative of a 2D Field

```
x = linspace(-5, 5., 100);
y = linspace(-5, 5., 100);
[X, Y] = meshgrid(x, y);

a = -0.2; b = 0.5; c=0.1;
d=0.5; e=0.; f=3.

dudx = 2*a*X + b + e*Y;

contourf(X, Y, dudx )
colorbar
```

$$\frac{\partial f}{\partial x}\Big|_{y \text{ constant}} = 2ax + b + ey$$



Partial Derivatives Numerical Solutions

- Previously we saw the first derivative for f=f(x) could be obtained from

$$\frac{df}{dx} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x} = \frac{f_{i+1}-f_i}{\Delta x}$$

- We calculate the derivatives numerically in the same way as used in ordinary derivatives (note the subscript notation is used again but in 2D with i and j)

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x,y)-f(x,y)}{\Delta x} \equiv \frac{f_{i+1,j}-f_{i,j}}{\Delta x}$$

$$\frac{\partial f}{\partial y} \approx \frac{f(x,y+\Delta y)-f(x,y)}{\Delta y} \equiv \frac{f_{i,j+1}-f_{i,j}}{\Delta y}$$

Second Order Numerical Partial Derivatives

- Previously we saw the second derivative for f=f(x) could be obtained from

$$\frac{d^2 f}{dx^2} \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{(\Delta x)^2}$$

- The numerical approximation for partial second derivative of f=f(x,y) is obtained in the same way,

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x+\Delta x, y) - 2f(x,y) + f(x-\Delta x, y)}{(\Delta x)^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2}$$

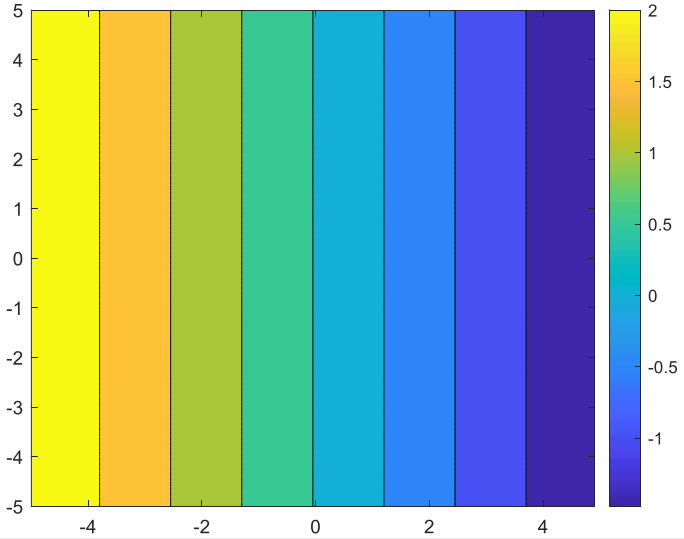
$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x, y+\Delta y) - 2f(x,y) + f(x, y-\Delta y)}{(\Delta y)^2} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2}$$

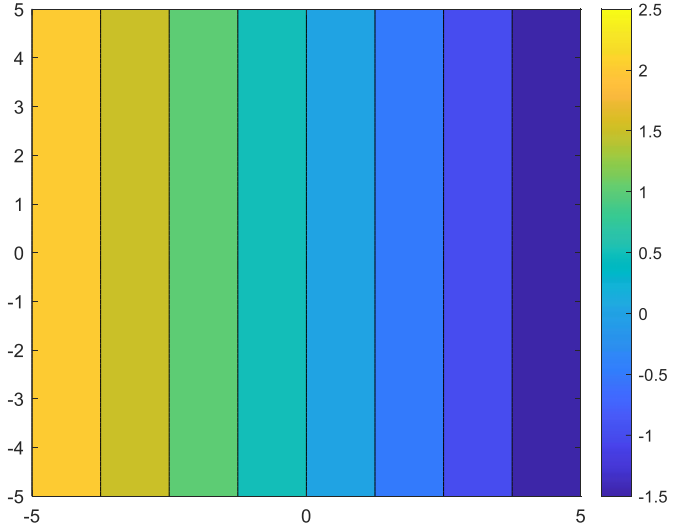
Numerical vs Analytical Derivatives in 2D

```
x = linspace(-5, 5., 100);
y = linspace(-5, 5., 100);
[X, Y] = meshgrid(x, y);

a = -0.2; b = 0.5; c=0.1;
d=0.5; e=0.; f=3.

u = a*X.^2 + b*X + c*Y.^2 ...
     + d*Y + e*X.*Y + f;
dudx_exact = 2*a*X + b + e*Y;
dx = X(1,2) - X(1,1);
dudx = diff(u,1,2)/dx

contourf(X, Y, dudx)
colorbar
```

$$\left.\frac{\partial f}{\partial x}\right|_{y \text{ constant}} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x}$$

$$2ax + b + ey$$





Partial Differential Equations

- To describe the change in fields, we use partial differential equations which vary in space (2D here), for example Laplace's Equation:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

- Often written using other notation,

$$\nabla^2 f = 0 \text{ or } \Delta f = 0 \text{ where } \nabla^2 \equiv \Delta \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

$$f_{xx} + f_{yy} = 0 \text{ where subscripts denote derivatives}$$

- In practice, fields are usually a function of three spatial coordinates and time (2D here for simplicity)

$$f = f(x, y, z, t)$$

Laplace's Equation

- Models the 2D diffusion in space using the sum of second order partial derivatives in x and y :

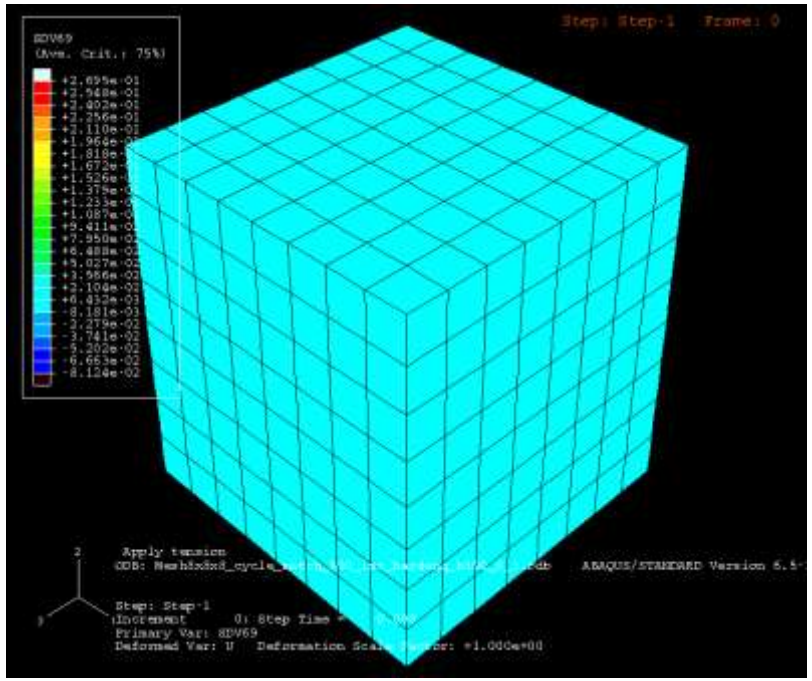
$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

- Note we have dropped the x=constant, y=constant for notational conciseness, but they are always implied by partial derivatives
- This equation describes the final state for the process of diffusion of a substance, such as ink in water, temperature in a metal, stress state in a material or concentration of a chemical in a mixture. It can also be solved to determine electromagnetic fields, potential fluid flow or gradients of pressure

Applications of Two and Higher Dimensional Equations

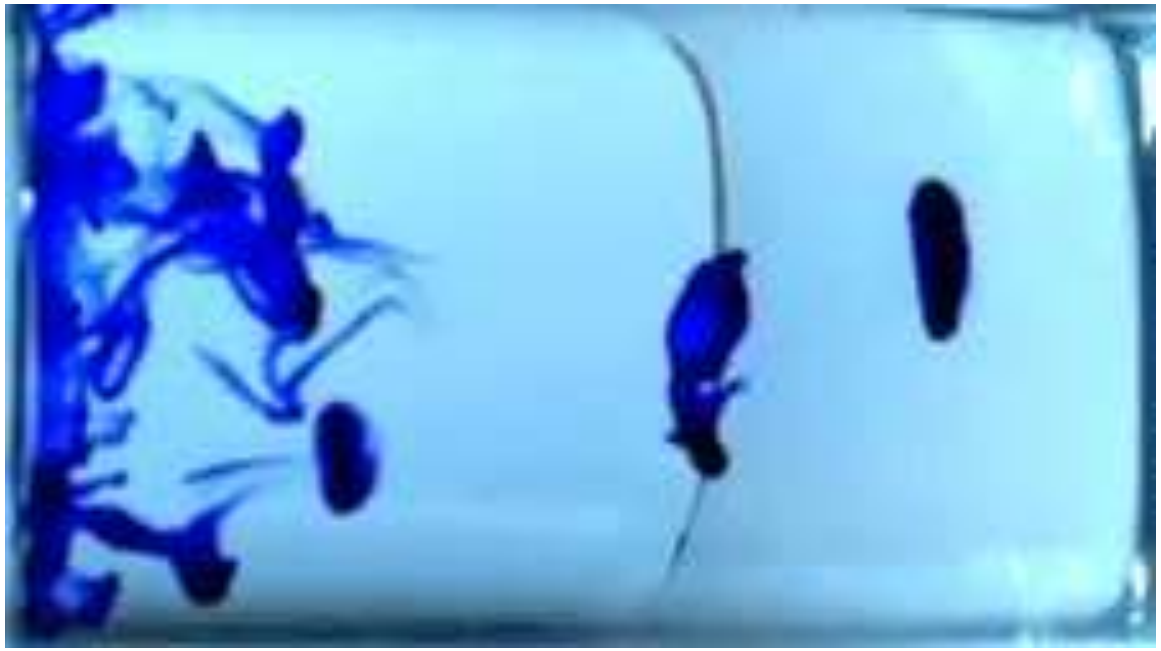
For heat conduction

For stress diffusion in solids

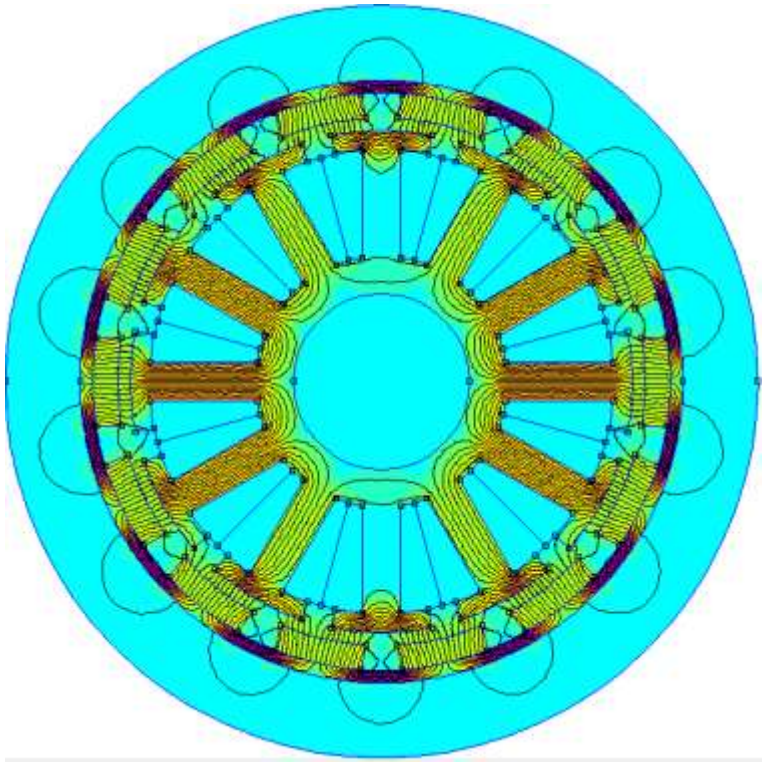


Applications of Two and Higher Dimensional Equations

For fluid dynamics



For electromagnetism



Solving the Laplace Equation Numerically

- There are a number of analytical solutions to this equation

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

- But we will use a numerical approximation for the second derivative, adapted for 2D,

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x+\Delta x, y) - 2f(x,y) + f(x-\Delta x, y)}{(\Delta x)^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2}$$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f(x, y+\Delta y) - 2f(x,y) + f(x, y-\Delta y)}{(\Delta y)^2} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2}$$

Solving the Laplace Equation Numerically

- So we are solving

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

$$\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2} = 0$$

- Which we rearrange to give

$$f_{i,j} = \frac{1}{2} \frac{(\Delta x)^2 (\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2} \left[\frac{f_{i+1,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} + f_{i,j-1}}{(\Delta y)^2} \right]$$

Simplifies to

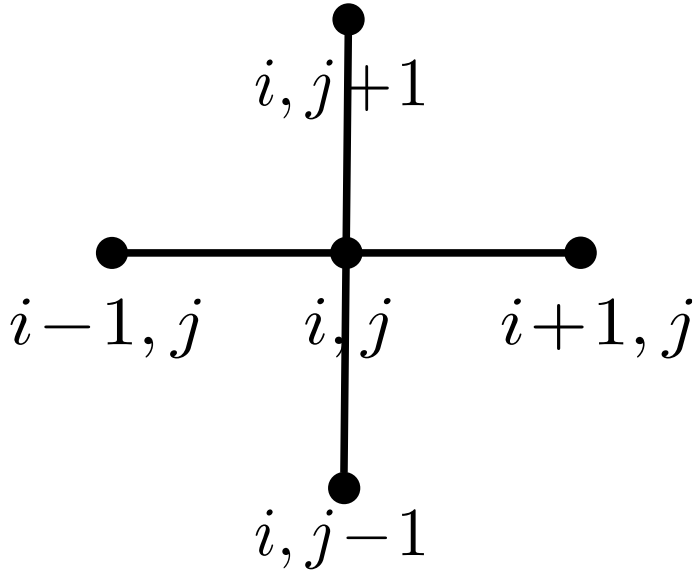
$$f_{i,j} = \frac{1}{4} \left[\frac{f_{i+1,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} + f_{i,j-1}}{(\Delta y)^2} \right] \Delta x = \Delta y = 1$$

Solving the Laplace Equation Numerically

- Using cell indices, derivatives in each direction can be seen to use what is called a five point “stencil”

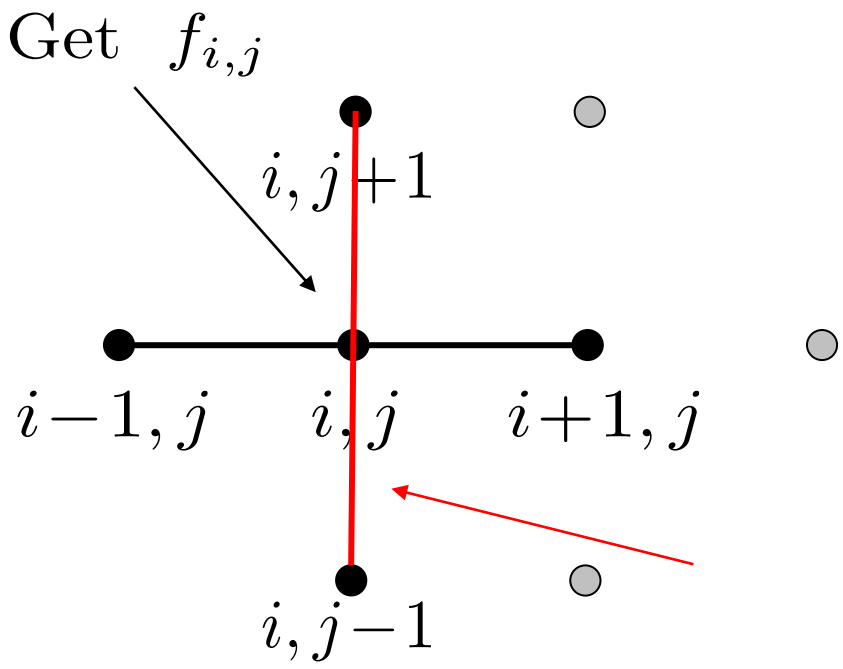
$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2}$$

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2}$$

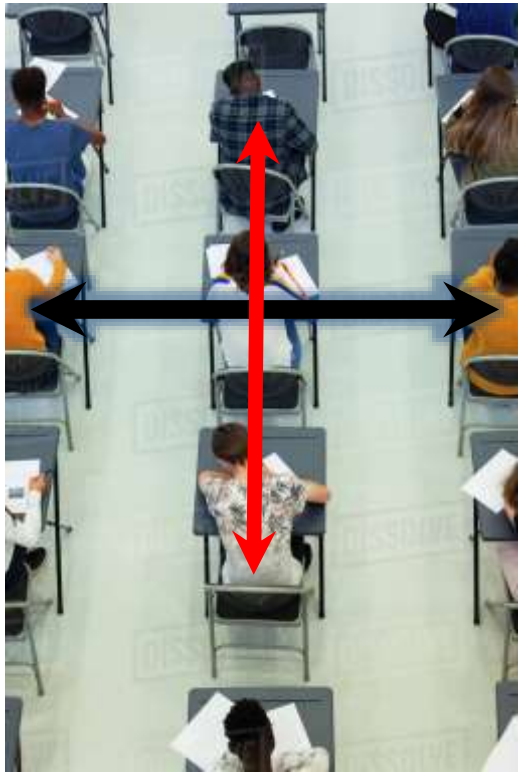


Solving 2D Finite Differences

- For a 2D grid we exchange points in both x and y

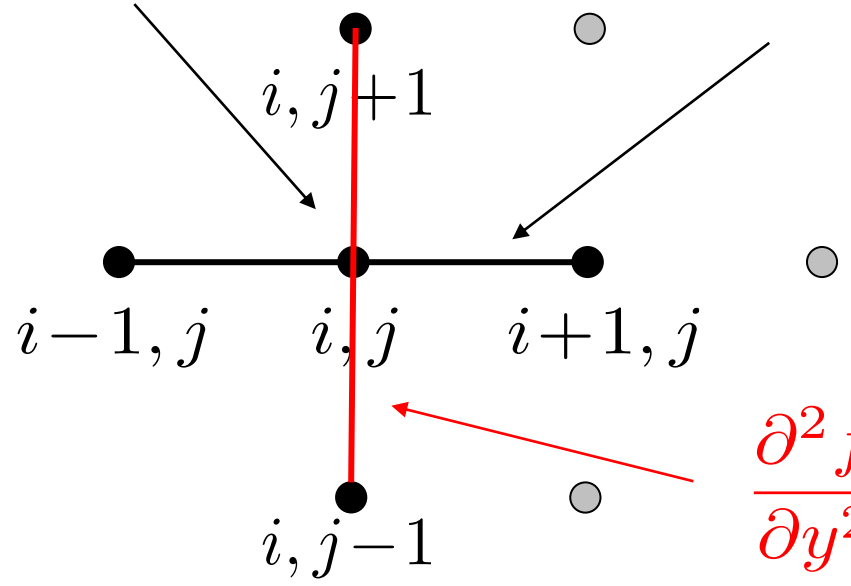


The first ever "CFD" used students to do the calculations and pass the results in 4 directions



Solving 2D Finite Differences

- For a 2D grid we exchange points in both x and y



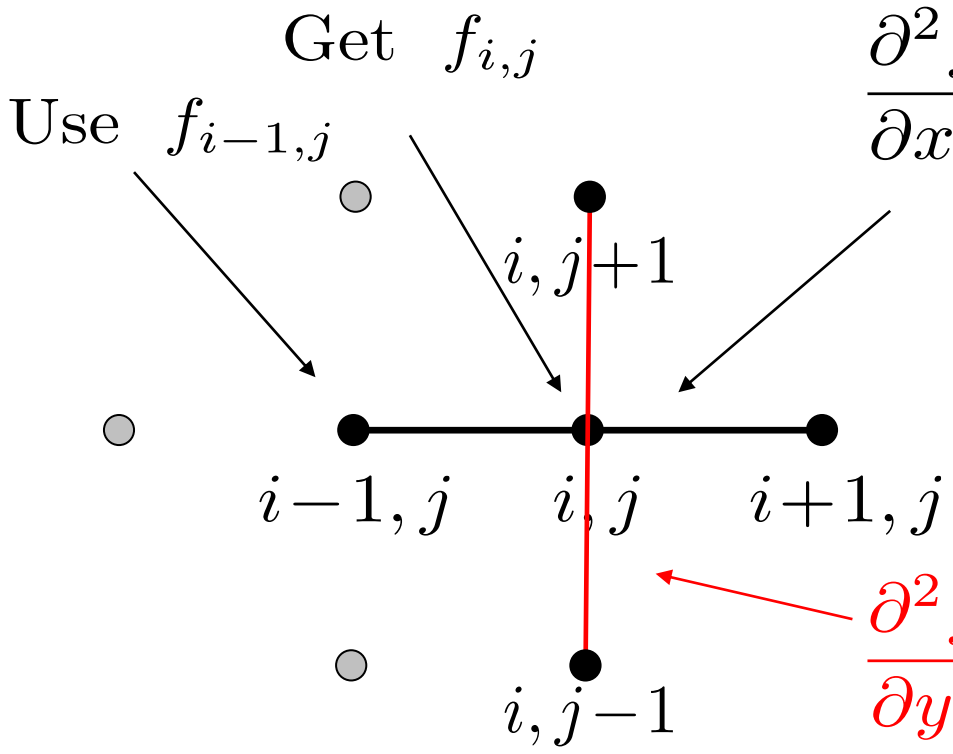
$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2}$$

Now we do in on a computer looping over i and j **(here i=1)**

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2}$$

Solving 2D Finite Differences

- Then we move to the next point



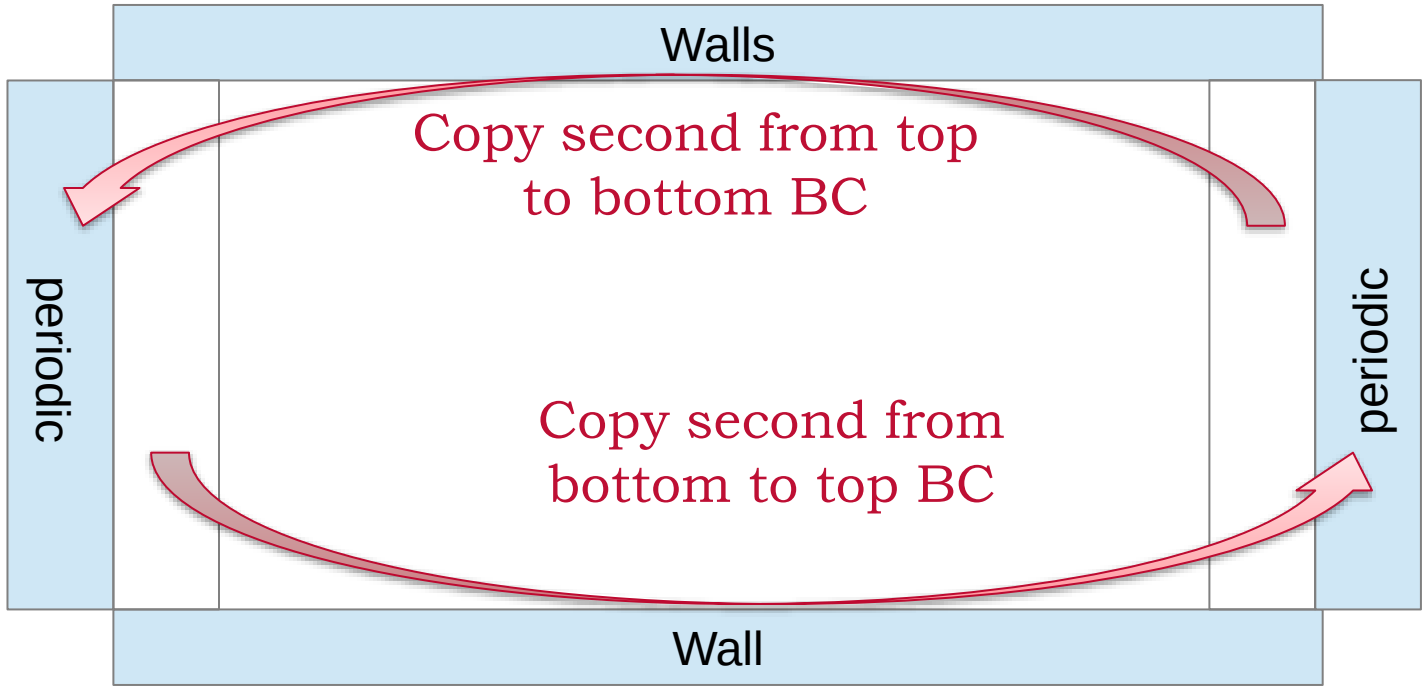
$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2}$$

Now we do in on a computer looping over i and j **(now i=2)**

$$\frac{\partial^2 f}{\partial y^2} \approx \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2}$$

Boundary Conditions

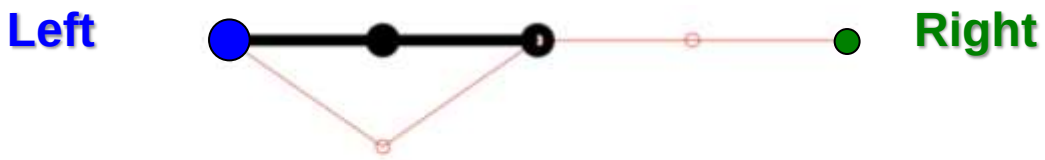
- We need to choose values for the 4 boundaries to model different physical cases, for example an infinite fluid between two plates



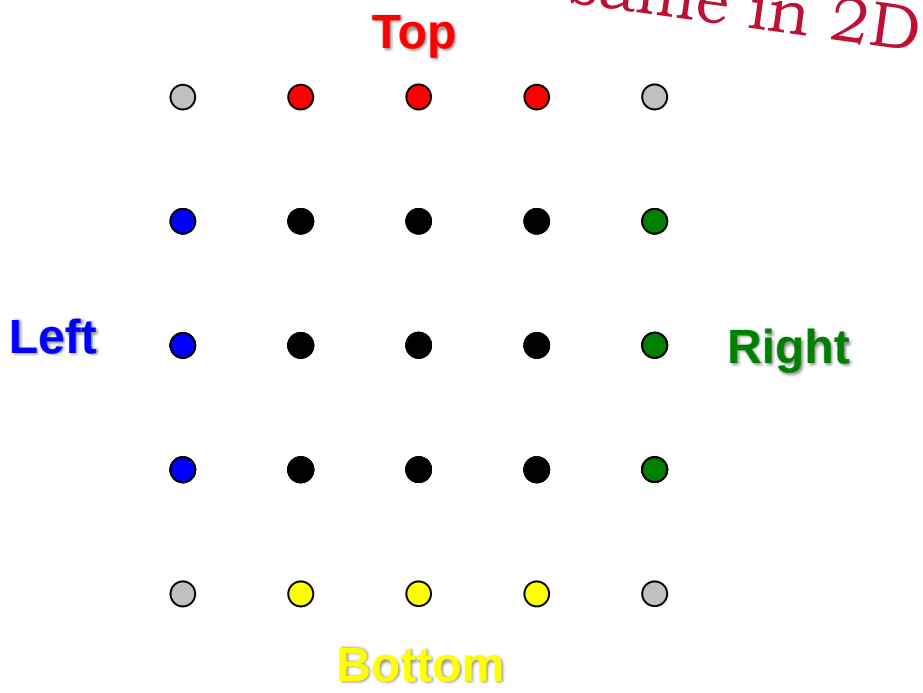
Solving by Iteration

- Recall in 1D example, we iterated `for iter=1:100` back and forth between the boundaries until we got to a solution which stops changing

Example in 1D



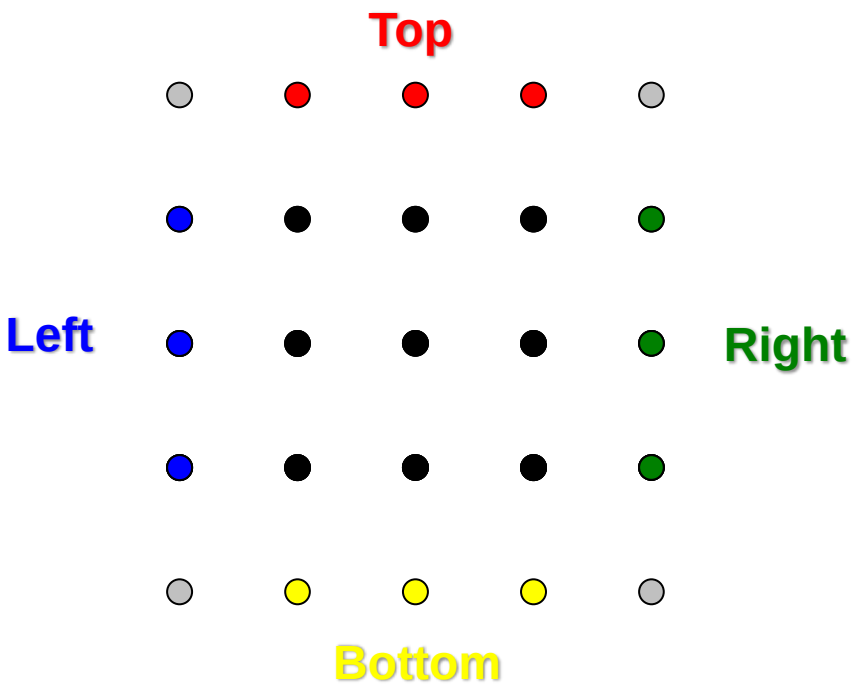
We do the same in 2D



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

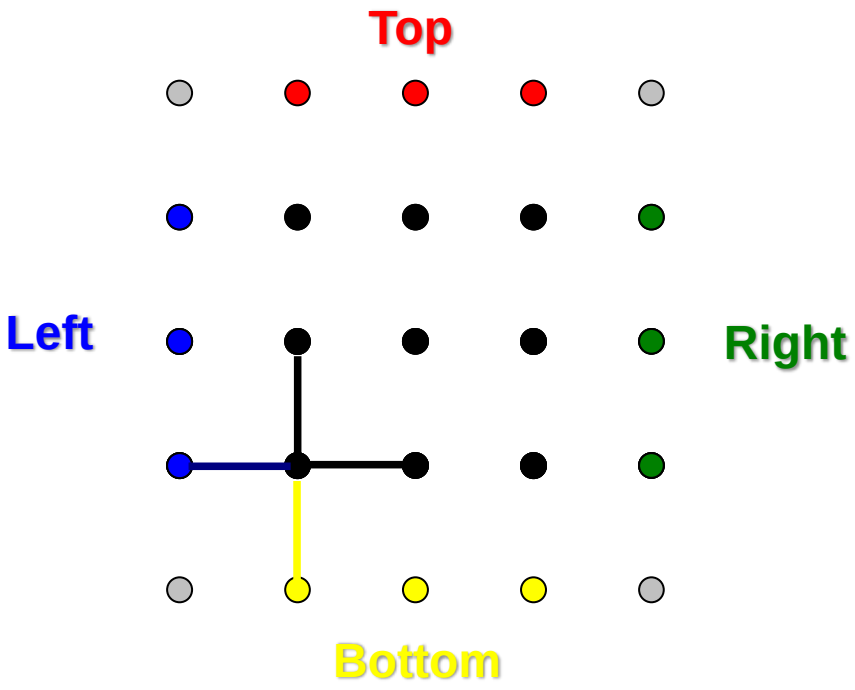
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

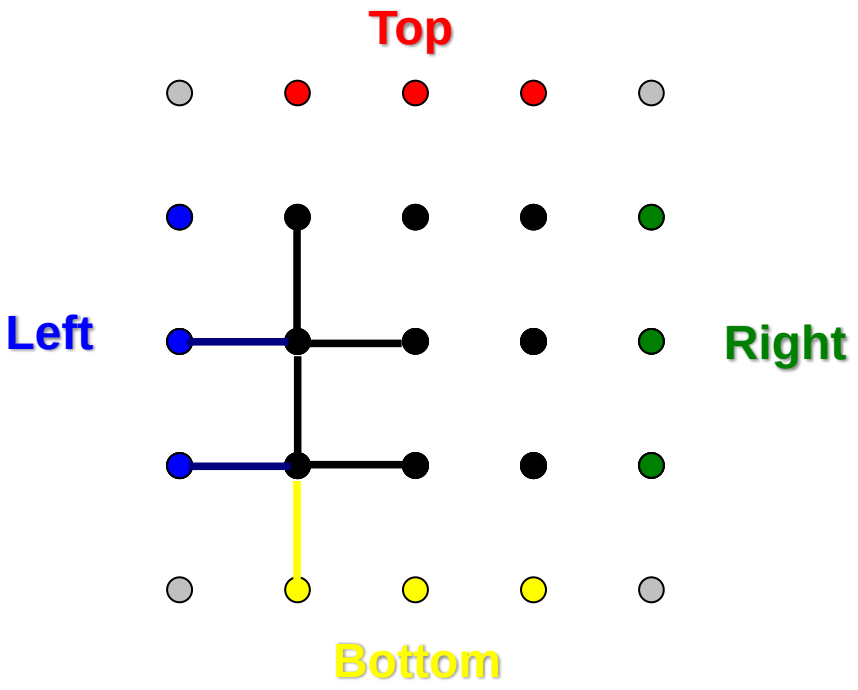
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

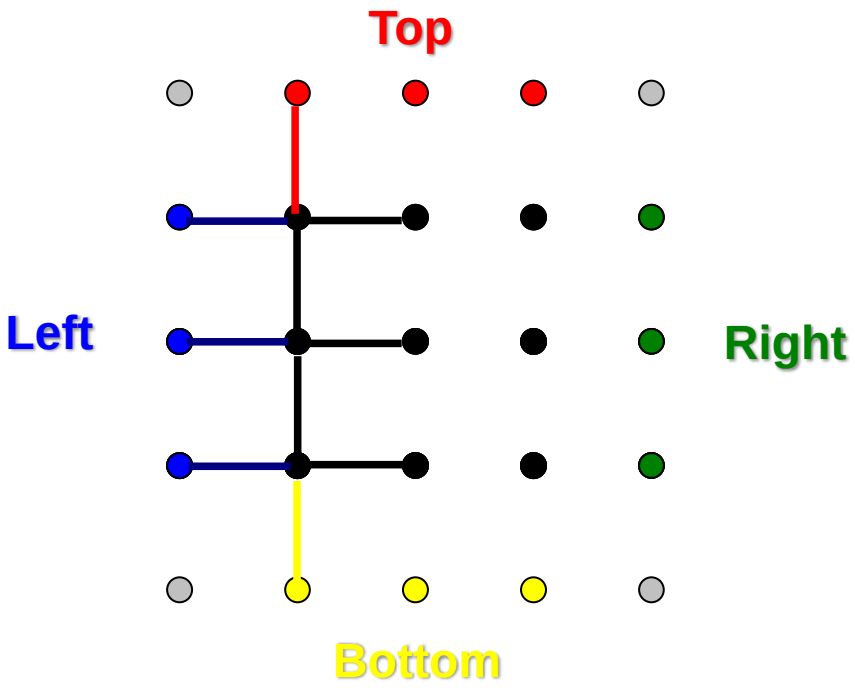
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

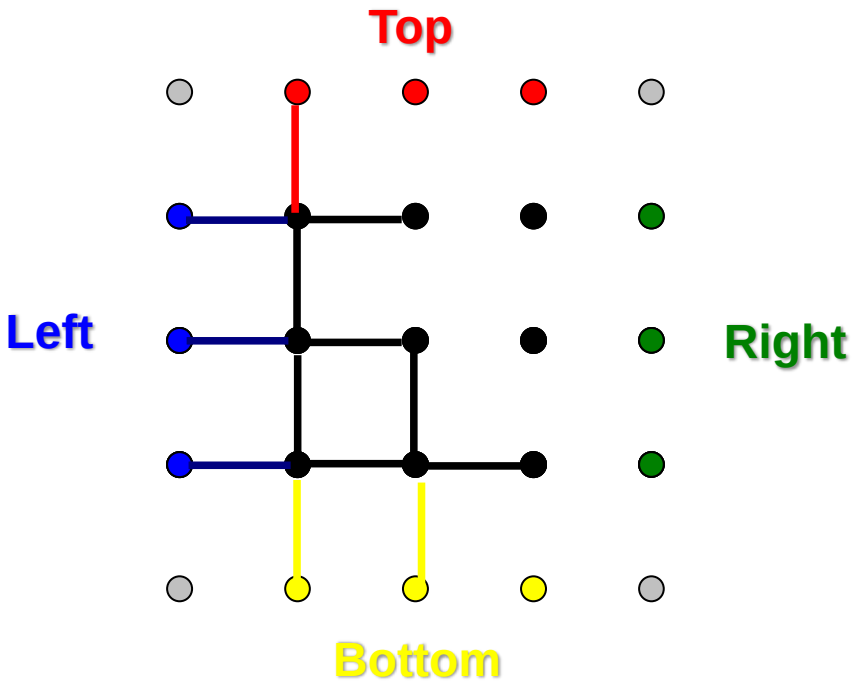
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

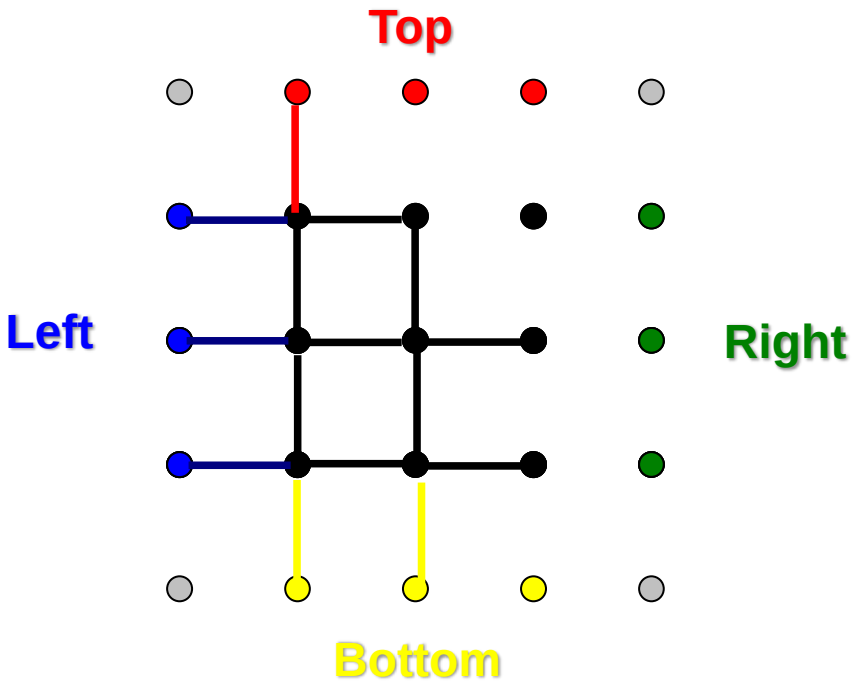
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

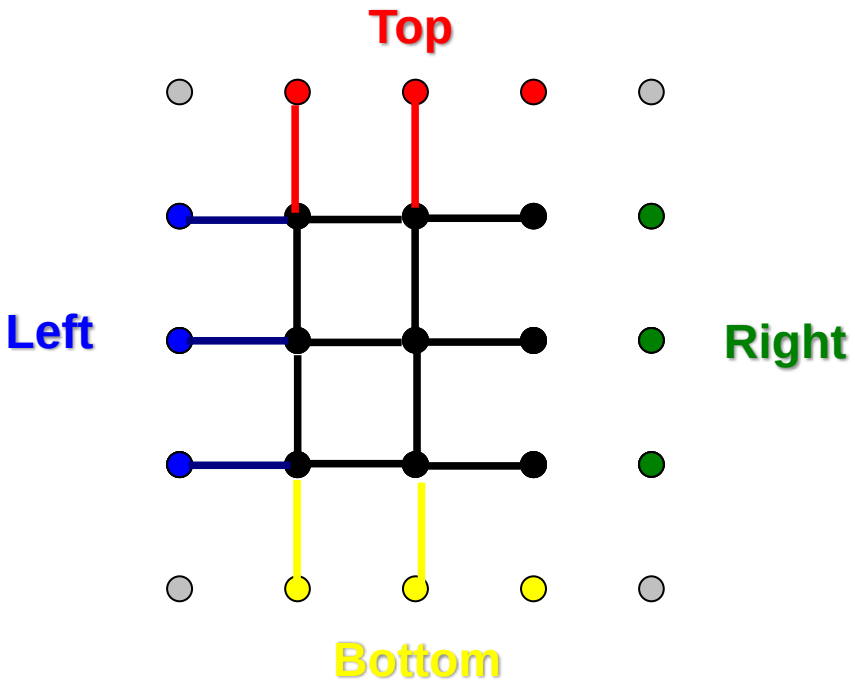
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

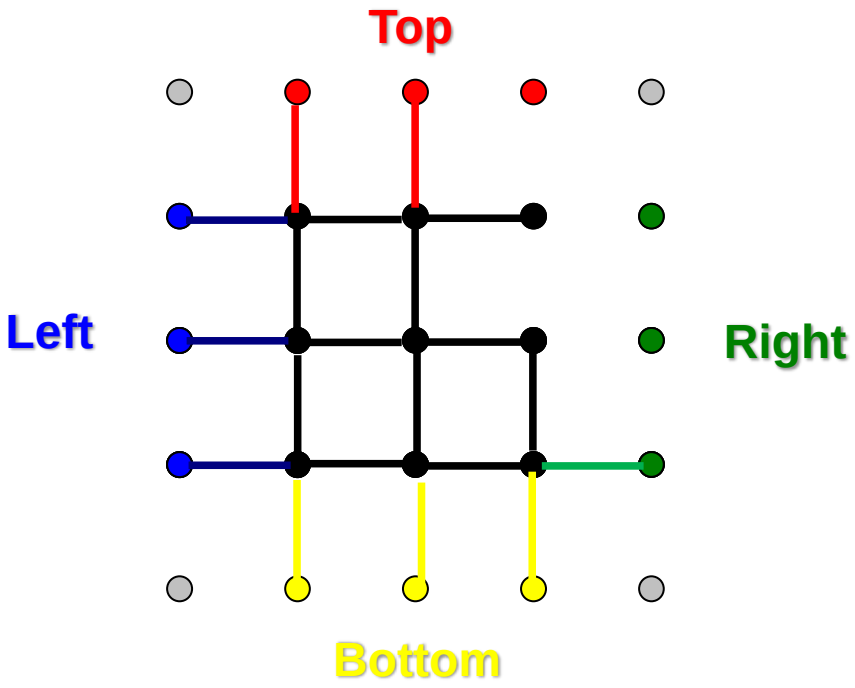
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

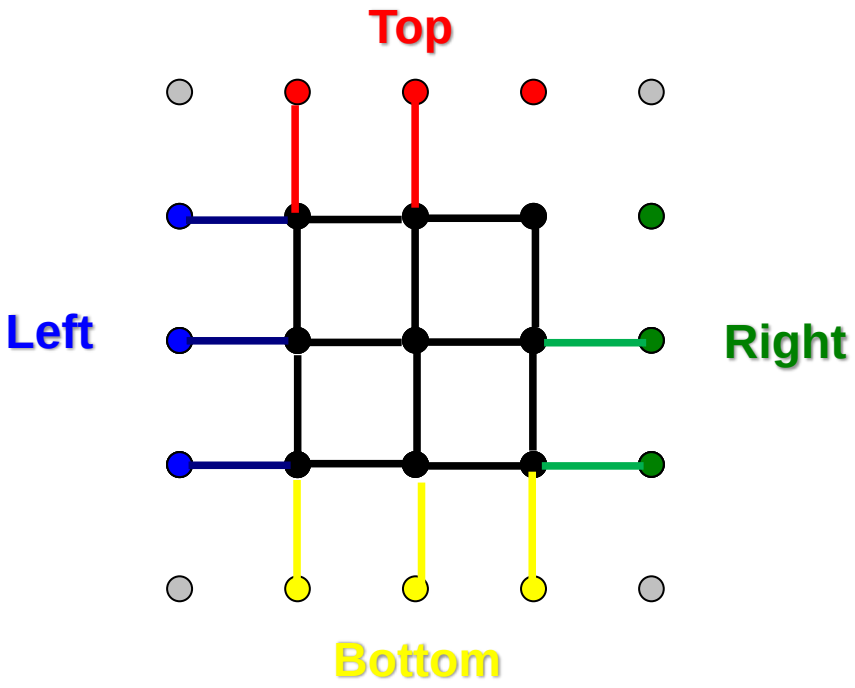
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

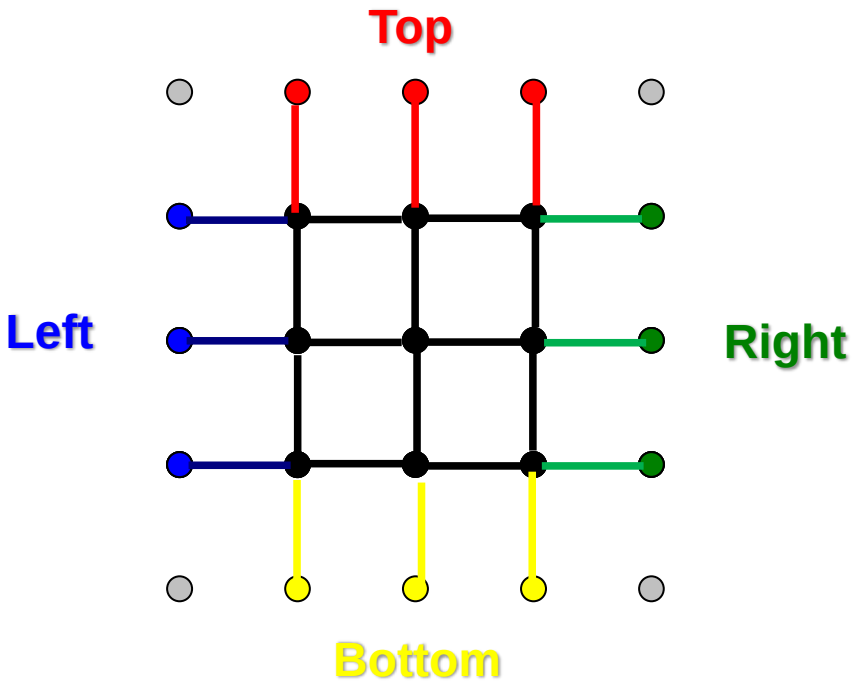
```
for i=2:M-1
    for j=2:M-1




    end
end
```



Solving by Iteration

- Notice we use points either side so start from the edge of our domain (boundary)
- Boundary conditions based on what we know about the flow

```
for i=2:M-1
    for j=2:M-1




    end
end
```

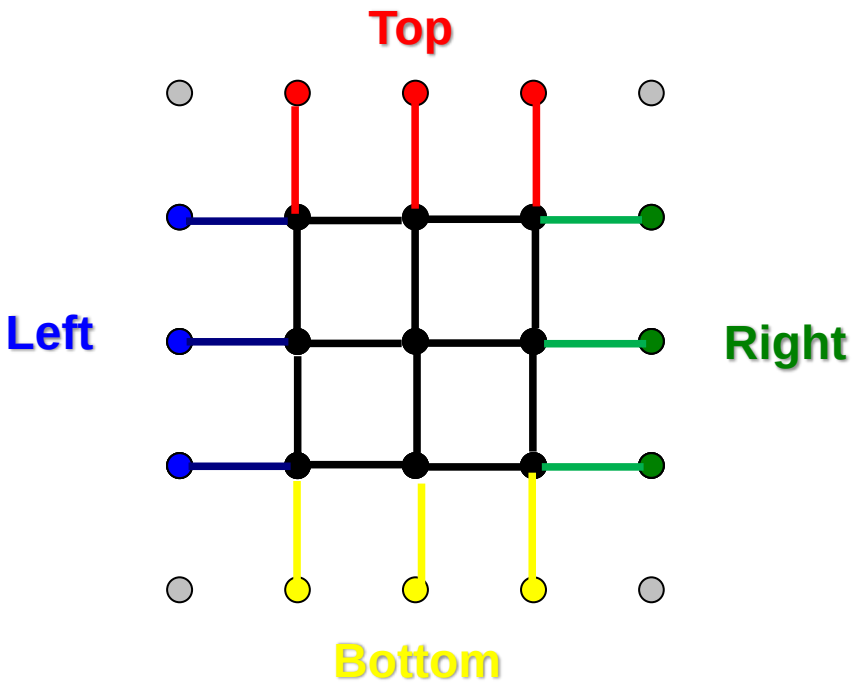


Solving by Iteration

- Iteration should proceeds until a solution is reached, convergence check: using a loop in MATLAB (like root finding)

$$\left| \sum_{i=1}^{9} \sum_{j=1}^{9} f_{i,j} - \sum_{i=1}^{9} \sum_{j=1}^{9} f_{i,j}^{\text{Previous Iteration}} \right| < \epsilon$$

```
for iter=1:100
    for i=2:M-1
        for j=2:M-1




        end
    end
end
```



Solving the Laplace Equation Numerically

- So we are solving

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

$$\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2} = 0$$

- Which we rearrange to give

$$f_{i,j} = C\left[\frac{f_{i+1,j} + f_{i-1,j}}{\Delta x^2} + \frac{f_{i,j+1} + f_{i,j-1}}{\Delta y^2}\right]$$

$$C = \frac{1}{2}\frac{(\Delta x)^2(\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}$$

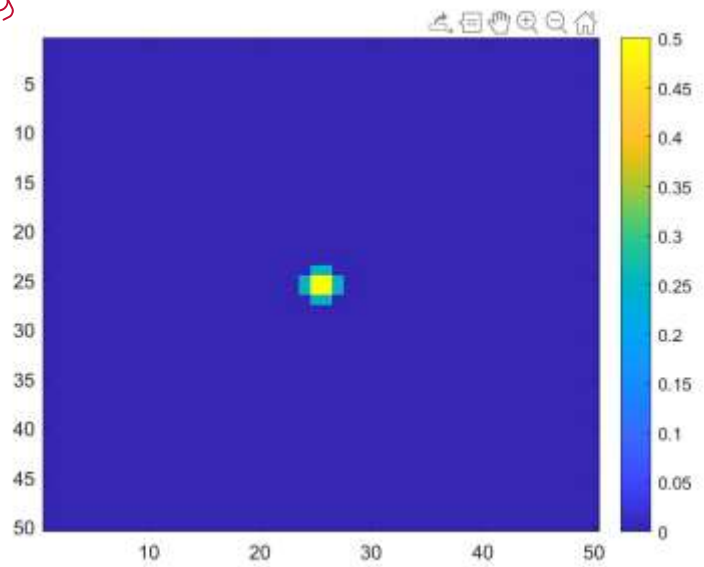
MATLAB 2D code

$$C = \frac{1}{2}\frac{(\Delta x)^2(\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}$$

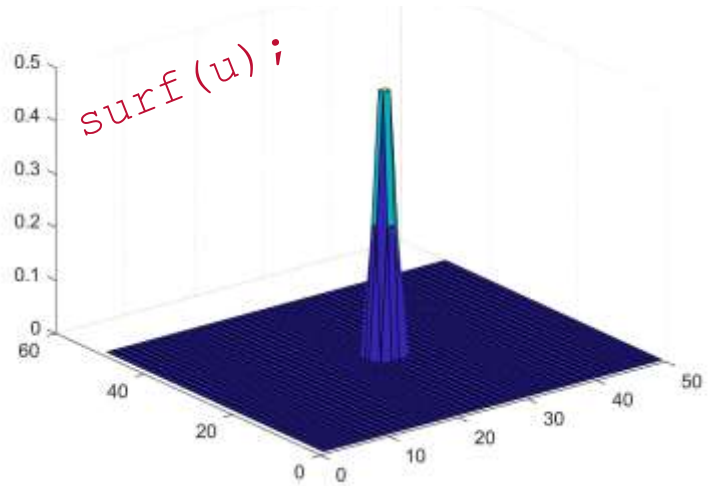
```
M = 50; Lx = 1; Ly = 1;
dx = Lx/(M-1); dy = Ly/(M-1);
u = zeros(M,M);
u(M/2:M/2+1, M/2:M/2+1) = 1; un=u;
C = (1/2)*dx^2*dy^2/(dx^2 + dy^2);
for it =1:100
    %Loop over all points
    for j=2:M-1
        for i=2:M-1
            un(i,j)=C*((u(i+1,j)+u(i-1,j))/dx^2 ...
                      +(u(i,j+1)+u(i,j-1))/dy^2);
        end
    end
    u = un;
    %Enforce Boundary Condition
    u(:,1) = u(:,end-1); %Bottom Wall Boundary
    u(1,:) = u(end-1,:); %Left periodic BC
    u(end,:) = u(2,:); %Right periodic BC
    u(:,end) = u(:,2); %Top Wall Boundary
 end
```

$$f_{i,j} = C\left[\frac{f_{i+1,j} + f_{i-1,j}}{\Delta x^2} + \frac{f_{i,j+1} + f_{i,j-1}}{\Delta y^2}\right]$$

imagesc(u);

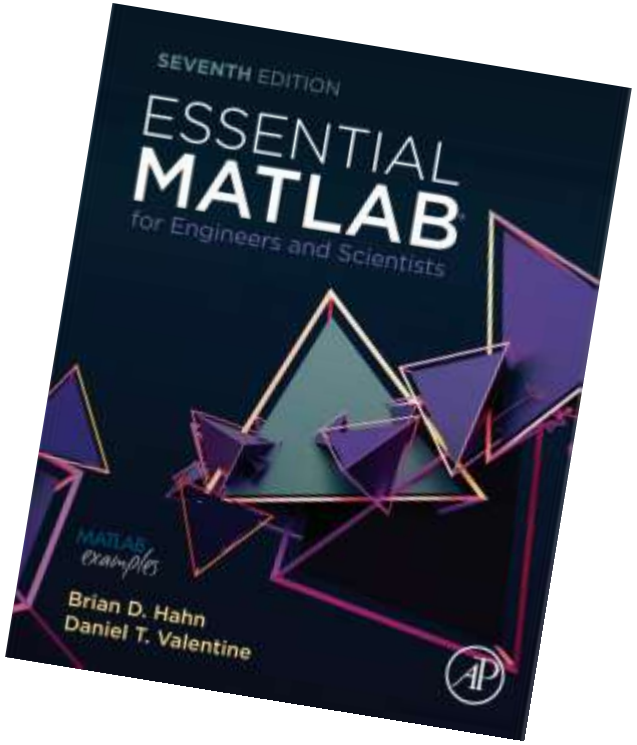


surf(u);



Summary

- Brief Recap
- Partial Differential Equation - Temporal spatial
  - Initial condition
  - Boundary conditions
  - Convective terms and forcing
- Partial Differential Equation - Two dimensional Laplace equations
  - Reminder of 2D fields and partial derivatives
  - Solution of Laplace's equation



Essential Matlab (EM)
http://tinyurl.com/yy53shga

Partial derivative example quite complex in EM as they use implicit

Other Boundary Conditions

- Direct (Dirichlet)
- Fluxes (Neumann)
- Mixed (Robin)
- Symmetry
- Periodic
  - Go out one side
  - Come back in other