

B1 Numerical Algorithms

4 Lectures, MT 2023

Lecture four – Computing Solutions of Partial Differential Equations.

Wes Armour

26th October 2023

Learning outcomes

➤ Linear PDE solution by finite differences (Lecture 4)

- Introduction,
- Elliptic (*Laplace*),
- Parabolic (*diffusion*),
- Hyperbolic (*wave*).



Pierre-Simon Laplace
1749 - 1827



Jean-Baptiste Joseph Fourier
1768 - 1830



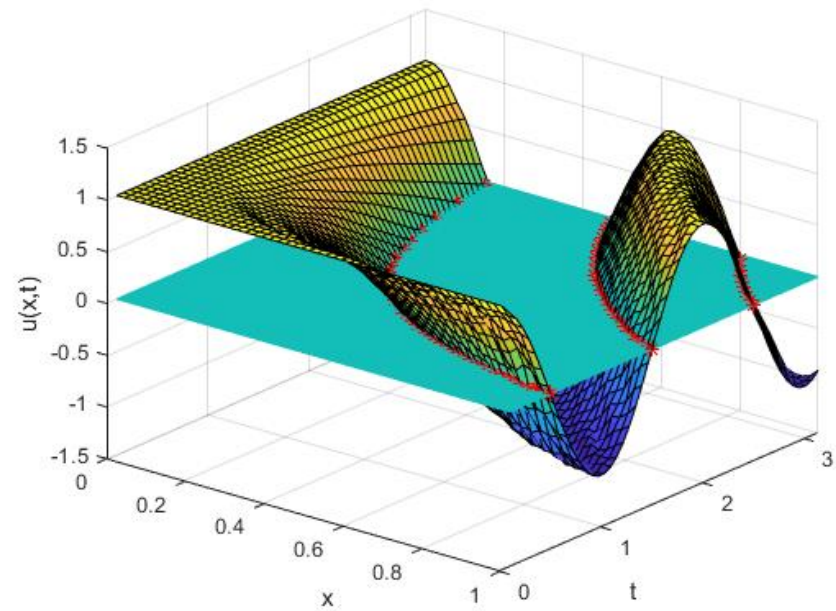
Siméon Denis Poisson
1781 – 1840

Partial Differential Equations (PDEs)

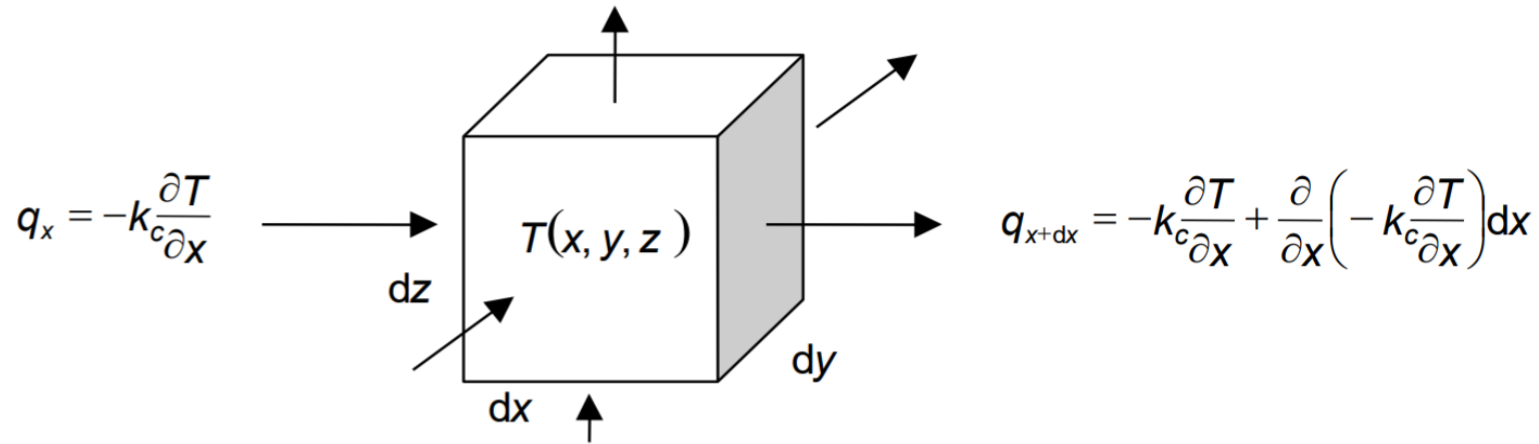
Partial differential equations (PDEs) arise in many engineering problems. Special cases can be solved exactly using analytic methods (see PDE lecture series notes).

However in practice, numerical methods are essential and much of Engineering rely on these techniques.

In this lecture, we consider the basics of solving PDEs using numerical methods, but for more sophisticated methods refer to textbooks (eg Burden & Faires Ch12).



Example of a PDE: Unsteady heat conduction



Consider the temperature T of an elemental cube $dx dy dz$ inside a body made of material conductivity k_c , density ρ , and specific heat capacity c .

There is a heat flux q_x into the y - z face etc. Then, as shown on the diagram, the total heat transfer into the cube in the x – direction is:

$$(q_x - q_{x+dx}) dy dz = \left\{ -k_c \frac{\partial T}{\partial x} - \left(-k_c \frac{\partial T}{\partial x} + \frac{\partial}{\partial x} \left(-k_c \frac{\partial T}{\partial x} \right) dx \right) \right\} dy dz = k_c \frac{\partial^2 T}{\partial x^2} dx dy dz$$

Example of a PDE: Unsteady heat conduction

The total heat transfer rate into the cube in all directions determines the rate at which it heats up:

$$k_c \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) dx dy dz = \rho c dx dy dz \frac{\partial T}{\partial t}$$

This gives the Unsteady Diffusion Equation in 3 dimensions, an example of a typical PDE:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = \frac{\rho c}{k_c} \frac{\partial T}{\partial t} = \nabla^2 T = \Delta T$$

Where $\Delta = \nabla^2$ is the Laplace operator.

Some other examples

Laplace's equation (e.g. steady state heat equation):

$$\nabla^2 T = 0$$

Poisson's equation (e.g. for electrostatics):

$$\nabla^2 \varphi = \rho(x, y, z)$$

The Wave Equation (you will already know this):

$$\nabla^2 V = c^2 \frac{\partial^2 V}{\partial t^2}$$

General form of second-order PDEs

The general form of a second-order PDE with two independent variables (x, y) is:

$$a(x, y)V_{xx} + b(x, y)V_{xy} + c(x, y)V_{yy} = F(x, y, V, V_x, V_y)$$

Where we have denoted the partial derivatives using subscripts:

$$V_x = \frac{\partial V}{\partial x} \qquad V_{xx} = \frac{\partial^2 V}{\partial x^2} \qquad V_{xy} = \frac{\partial^2 V}{\partial x \partial y}$$

General form of second-order PDEs

PDE's are classified according to the values of a, b, c :

Equation Type:	If	Example	Applications
Elliptic	$ac - b^2 > 0$	Laplace Equation $V_{xx} + V_{yy} = 0$	Steady Heat conduction. Electrostatics. Fluid flow
Parabolic	$ac - b^2 = 0$	Diffusion equation $V_{xx} = V_t$	Unsteady heat conduction. Water seepage. EM fields in conductors
Hyperbolic	$ac - b^2 < 0$	Wave equation $V_{xx} = V_{tt}$	Waves Fluid flow

It's important to note that because the coefficients a, b, c are all functions of the variables (x, y) , they may change value in different parts of the solution space.

Hence equations may change type in different regions of the solution space.

Working with PDEs



The Euler equations which describe compressible fluid flow are **elliptic** if the flow is **subsonic**, but **hyperbolic** if it is **supersonic**.

This can happen in the flow-field around an aeroplane at speeds just less than Mach 1 (the speed of sound).

Solutions for the subsonic (elliptic) and supersonic (hyperbolic) zones have to be obtained by different methods and patched together!

Working with PDEs



A vapour cone is a visible cloud of condensed water that can form around an object moving at high speed through moist air.

When the local air pressure around the object drops, so does the local air temperature.

If the temperature drops below the saturation temperature, a cloud forms.

This phenomena typically occurs around aircraft travelling at transonic speeds

So, very complicated physics to model!

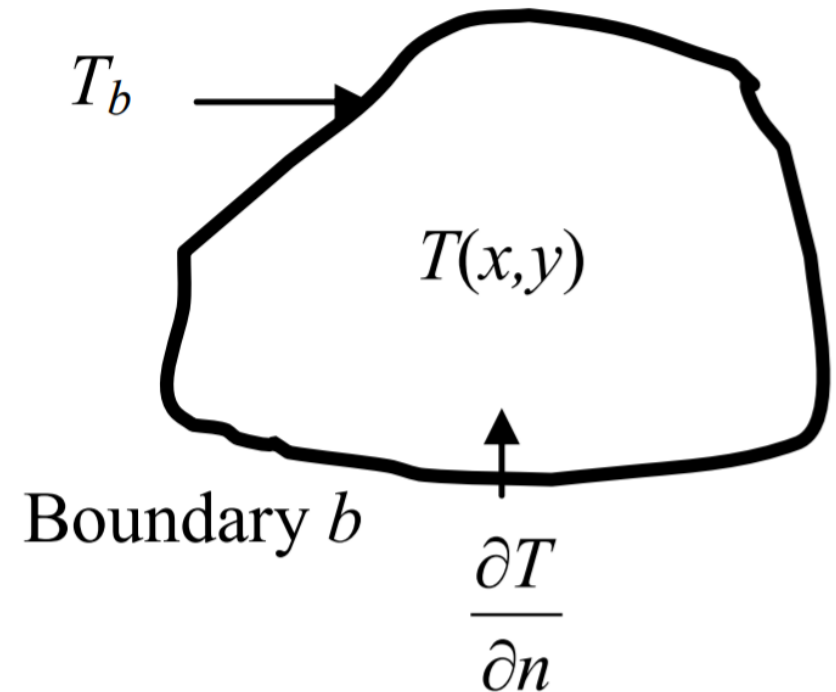
Boundary Conditions for Second-order PDEs

Generally, the solution area of a PDE is surrounded by a boundary, the conditions on which determine the solution.

It is a **Dirichlet** or *first boundary value problem* if T_b is specified on the boundary.

It is a **Neumann** or *second boundary value problem* if $\frac{\partial T}{\partial n}$, the normal derivative is specified on the boundary.

It is a Mixed boundary value problem if T_b and $\frac{\partial T}{\partial n}$, are specified on different parts of the boundary.



The general approach

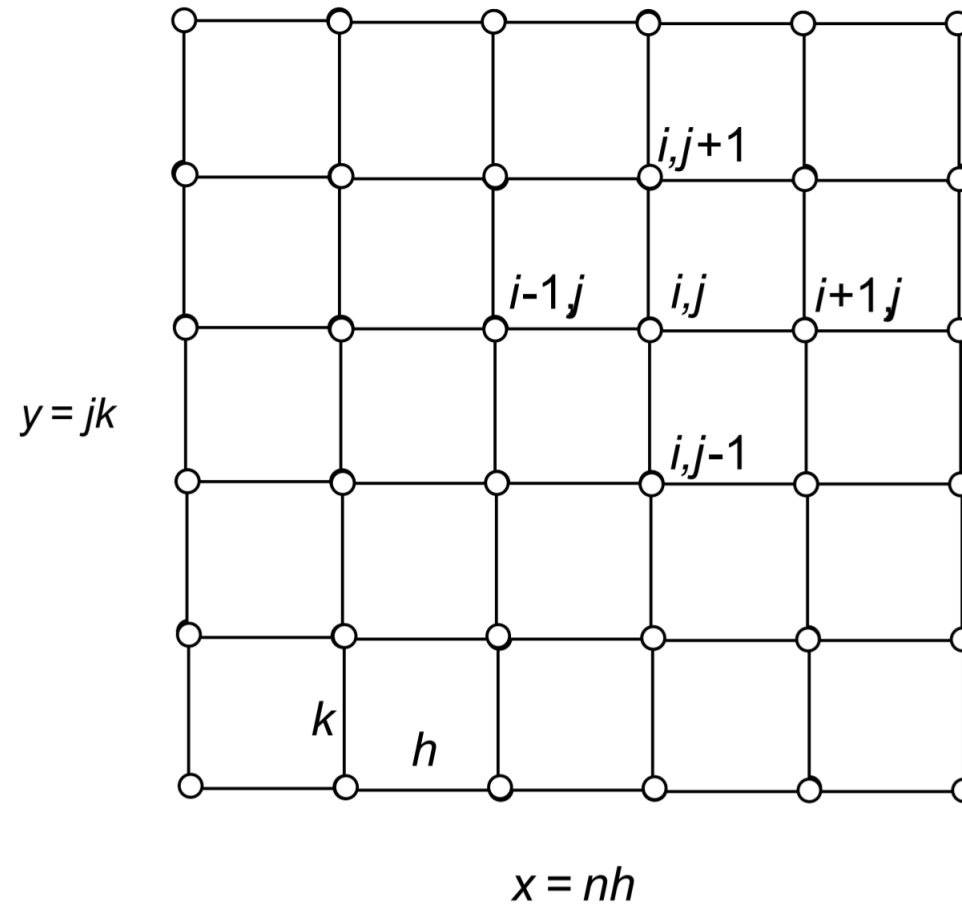
Approximate the continuous solution domain by a discrete set of points

$$x = ih, \quad y = jk, \quad i = 1, \dots, M, \quad j = 1, \dots, N$$

spaced at intervals h and k .

Note: we will start indexing at 1, since MATLAB arrays start at 1. Many books start at 0!

Then derive a suitable discrete approximation to the PDE and choose a suitable technique to solve it, subject to the boundary conditions.



Forward difference

Recall our definition of the derivative:

$$f' = \frac{df}{dt} = \lim_{\delta t \rightarrow 0} \frac{f(t + \delta t) - f(t)}{\delta t}$$

We can define the **forward difference** approximation as:

$$f'_n \approx \frac{f_{n+1} - f_n}{h} = \frac{f(nh + h) - f(nh)}{h}$$

Where in our case we have a barometer sensor that is sampled at discrete time intervals of length h (5 seconds). Then the value of the function at point n will be given by $f_n = f(nh) = f(5n)$.

Returning to our Taylor series:

$$f(x + \delta x) = f(x) + \delta x \frac{df(x)}{dx} + \frac{\delta x^2}{2!} \frac{d^2 f(x)}{dx^2} + \frac{\delta x^3}{3!} \frac{d^3 f(x)}{dx^3} + \dots,$$

And rearranging we have:

$$\underbrace{\frac{f(x+\delta x) - f(x)}{\delta x}}_{\text{What we want}} = \frac{df(x)}{dx} + \underbrace{\frac{\delta x}{2!} \frac{d^2 f(x)}{dx^2} + \frac{\delta x^2}{3!} \frac{d^3 f(x)}{dx^3} + \dots}_{\text{Error}},$$

Hence the forward difference approximation has an error that will scale with δx – because this is the largest value in the error term!

So the error is “Order” $O(\delta x) = h = \delta t$

Formulae for higher order derivatives.

It is also possible to construct difference equations that approximate higher order derivatives.

Consider the second order central difference:

$$f''(x) = \frac{\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}}{h} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

Found quite simply by applying the central difference scheme to the first derivative...

Using Taylor series expansion (as before) show that the error is $O(h^2)$

Formulae for more than one variable.

It's possible to estimate partial derivatives for more than one variable. Here are some useful formulae:

$$f_x(x, y) = \frac{f(x + h, y) - f(x - h, y)}{2h}$$

$$f_y(x, y) = \frac{f(x, y + k) - f(x, y - k)}{2k}$$

$$f_{xy}(x, y) = \frac{f(x + h, y + k) - f(x + h, y - k) - f(x - h, y + k) + f(x - h, y - k)}{4hk}$$

Implicit vs Explicit Methods

Often you will hear a certain method described as Implicit or Explicit.

An Explicit method calculates the state of a system at a later step (e.g. time) from the state of the system at the current step (time).

$$x_{t+1} = f(x_t)$$

An Implicit method finds a solution by solving an equation involving both the current state of the system and the later state.

$$F(x_t, x_{t+1}) = 0$$

Solving for x_{t+1}

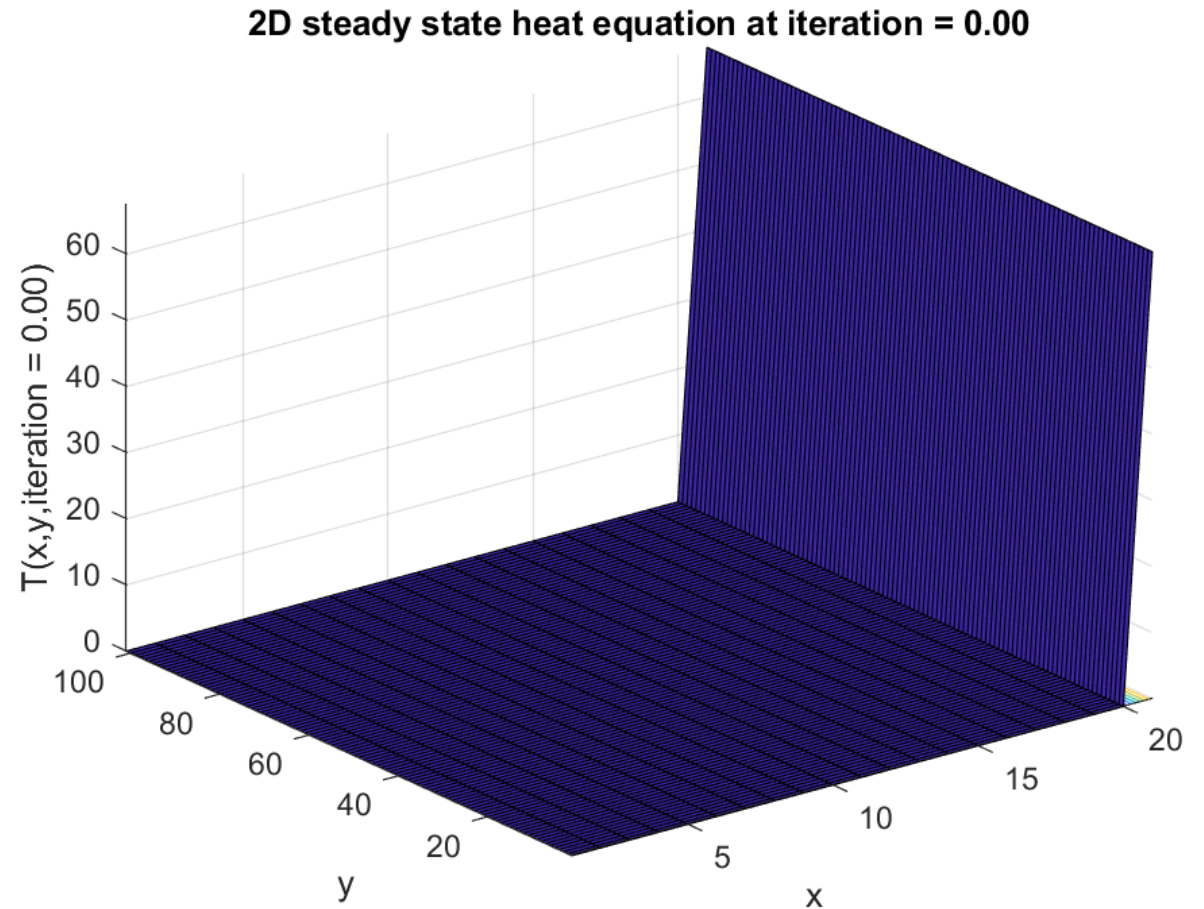
Part B – Numerical Solution
of Elliptic Equations

An example problem

Solve Laplace's Equation to obtain the steady state temperature in the rectangular region $x = 1$ to 20 , $y = 1$ to 100 , subject to the boundary conditions:

$$T(20, y) = [67.5]$$
$$T(1, y) = T(x, 1) = T(x, 100) = [0]$$

So a constant temperature of 67.5C along the edge where $x = 20$, all other boundaries set to zero.



Solution

Approximate the second derivatives in the 2-D Laplace's equation using the central difference formula from lecture 1:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{k^2} + O(h^2) + O(k^2)$$

Next set $k = h$ (so we are working on a square mesh), and re-arrange to give:

$$T_{i,j} = \frac{1}{4} (T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}) + O(h^4)$$

Solving the discrete approximation equation

For fixed boundary conditions, this is a set of $(M - 2)(N - 2)$ simultaneous equations, $(100 - 2)(20 - 2) = 1764$ in this example.

Small examples could be solved completely (directly) by, say, Gaussian elimination. However, for our example direct solution would be computationally costly!

Instead, iterate by applying our discrete equation to update all points in the solution area one step at a time.

Seen this before? It is very similar to the Jacobi algorithm in A1.

This is an Implicit Method since the values at the point (i, j) depend on the four surrounding points, and we must update all the values at each iteration stage.

Error analysis: It can be shown to converge with an error $O(h^2 + k^2)$


Numerical Solution of Laplace Equation

Panopto bonus video

B1 Numerical Algorithms

4 Lectures, MT 2022
Lecture four.
Bonus video 1 – Numerical Solution of Laplace Equation
Wes Armour

Errors/typos/questions to: wes.armour@ecs.qul.ac.uk



1

★ 00:20


Solution

Approximate the second derivatives in the 2-D Laplace's equation using the central difference formula from lecture 1:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \Rightarrow \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{k^2} + O(h^2) + O(k^2)$$

Next set $k = h$ (so we are working on a square mesh), and re-arrange to give:

$$T_{i,j} = \frac{1}{4}(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}) + O(h^4)$$



2

★ 05:31

Solving the discrete approximation equation

For fixed boundary conditions, this is a set of $(M-2)(N-2)$ simultaneous equations, $(100-2)(20-2) = 1764$ in this example.

Small examples could be solved completely (directly) by, say, Gaussian elimination. However, for our example direct solution would be computationally costly!

Instead, iterate by applying our discrete equation to update all points in the solution area one step at a time.

Seen this before? It is very similar to the Jacobi algorithm in A1.

This is an implicit Method since the values at the point (i,j) depend on the four surrounding points, and we must update all the values at each iteration stage.

Error analysis: It can be shown to converge with an error $O(h^2 + k^2)$



3

★ 01:32

The end...



4

★ 00:16

```
%Dimensions of the simulation grid in x (xdim) and y (ydim) directions
```

```
xdim=100; ydim=20;
```

```
% Error cutoff
```

```
cutoff=0.05;
```

```
%Initializing previous (T_prev) and present (T_now) temp matrices
```

```
T_now=zeros(xdim+1,ydim+1);
```

```
T_prev=zeros(xdim+1,ydim+1);
```

```
%Initializing boundary conditions only for T_now
```

```
i=1:1:xdim+1;%x-co-ordinates for boundary at y=ydim*grid_size
```

```
%A constant temperature of 100C applied to one boundary
```

```
T_prev(i,ydim+1)=67.5;
```

```
error=max(max(abs(T_now-T_prev)))
```

```
figure;
```

```
t=0;
```

```
while error > cutoff
```

```
    for i=2:1:xdim
```

```
        for j=2:1:ydim
```

```
            T_now(i,j)=(T_prev(i+1,j)+T_prev(i-1,j)+T_prev(i,j+1)+T_prev(i,j-1))/4.0;
```

```
        end
```

```
    end
```

```
    error=max(max(abs(T_now-T_prev)))
```

```
    T_prev=T_now;
```

```
    surfc(T_now);
```

```
    title(sprintf('2D steady state heat equation at iteration = %1.2f',t),'FontSize',11);
```

```
        xlabel('x','FontSize',11); ylabel('y','FontSize',11);
```

```
        zlabel(sprintf('T(x,y,iteration = %1.2f)',t),'FontSize',11);
```

```
    drawnow;
```

```
    t=t+1;
```

```
end
```

MATLAB Solution

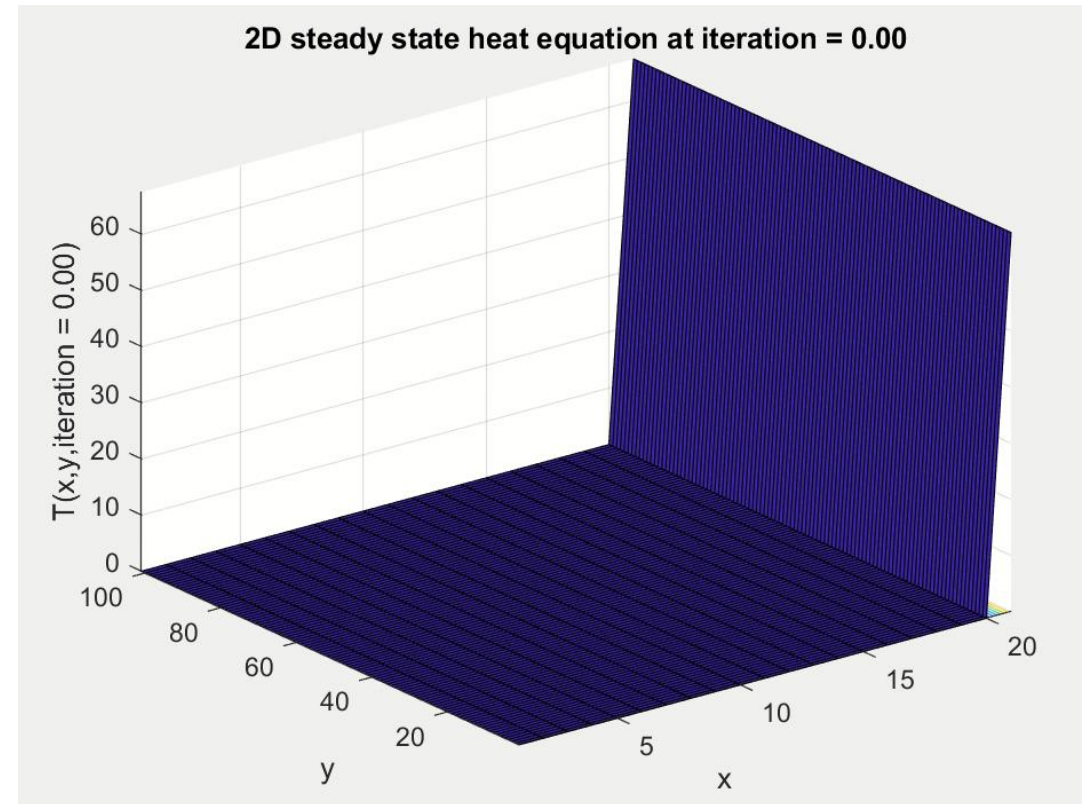
Final solution surface

Successive iterations move towards the final solution.

The code exits when the difference between iteration i and $i + 1$ is smaller than a predefined cut-off value.

Taking an initial “guess” at the solution can speed up the computation.

For large systems a method such as Successive-Over-Relaxation is often used in practice as this has good resistance to round-off errors.



Part C – Numerical Solution of Parabolic Equations

Numerical methods for Parabolic PDEs

Again, approximate the continuous solution domain by a discrete set of points:

$$x = ih, \quad t = jk$$

With:

$$i = 1, \dots, LM \quad j = 1, \dots, LN$$

And spacing h, k .

Then derive a suitable discrete approximation to the PDE and choose a suitable technique to solve it, subject to the boundary conditions.

An example problem

Solve Diffusion Equation to obtain the transient temperature through a one-dimensional region $x = 1$ to 10 , subject to the boundary conditions:

$T(x, 1) = 0\text{ }^{\circ}\text{C}$, i.e. the whole rod is cold ($0\text{ }^{\circ}\text{C}$) at the starting time.

$T(1, t) = 100\text{ }^{\circ}\text{C}$, i.e. the end of the rod is then held at $100\text{ }^{\circ}\text{C}$ for all time

In other words, the temperature in a bar which has one end suddenly raised by $100\text{ }^{\circ}\text{C}$ from an initial condition of the whole rod having a temperature of 0°C .

Solution

Again we find a discrete approximation equation.

In this case, we can use a **central difference approximation in space x** but a **forward difference in time t** :

So our diffusion equation becomes:

$$\frac{\partial^2 T}{\partial x^2} = \alpha \frac{\partial T}{\partial t}$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2} + O(h^2) = \alpha \left(\frac{T_{i,j+1} - T_{i,j}}{k} \right) + O(k)$$

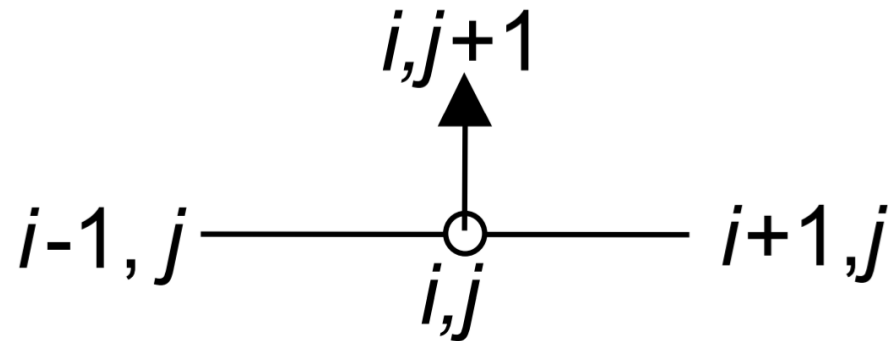
Rearranging this we find:

$$T_{i,j+1} = T_{i,j} + \frac{k}{\alpha h^2} (T_{i+1,j} - 2T_{i,j} + T_{i-1,j}) + O(k^2) + O(kh^2)$$

Solving the discrete approximation equation

This is an Explicit formula which gives the temperatures in row $j + 1$ in terms of row j .

Thus the calculation can be progressed row by row from the initial condition, $T(x, 1)$, and the changes of T with time t can be calculated directly in a progressive fashion.



Error Analysis

Of course, the error $O(k^2) + O(hk^2)$ different in the x and t direction, and the accuracy of the final result may depend on a suitable choice of $r = \frac{k}{\alpha h^2}$ which is the **update weighting factor**.

Analysis shows that the method converges if $r \leq \frac{1}{2}$ (i.e. the scheme is **conditionally stable**)

MATLAB Solution

Code snippet:

```
for j=1:TN-1 %TN -1 time steps
    for i = 2:TM-1 % calculate a new column
        T2(i,j+1) = T2(i,j) + r*(T2(i+1,j)+ -2*T2(i,j) + T2(i-1,j));
    end
end
```

Gives:

100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
0	40	48	56	60	64	66	69	70	72	73	74	75	76	77	78	78	79	80	80
0	0	16	22	29	34	38	42	45	47	49	51	53	54	56	57	58	59	60	61
0	0	0	6	10	15	18	22	25	28	30	32	34	36	38	39	41	42	44	45
0	0	0	0	3	5	7	10	12	14	16	19	20	22	24	26	27	28	30	31
0	0	0	0	0	1	2	3	5	6	8	10	11	13	14	15	17	18	19	20
0	0	0	0	0	0	0	1	2	2	3	4	5	6	7	9	10	11	12	13
0	0	0	0	0	0	0	0	0	1	1	2	2	3	4	4	5	6	6	7
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	2	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

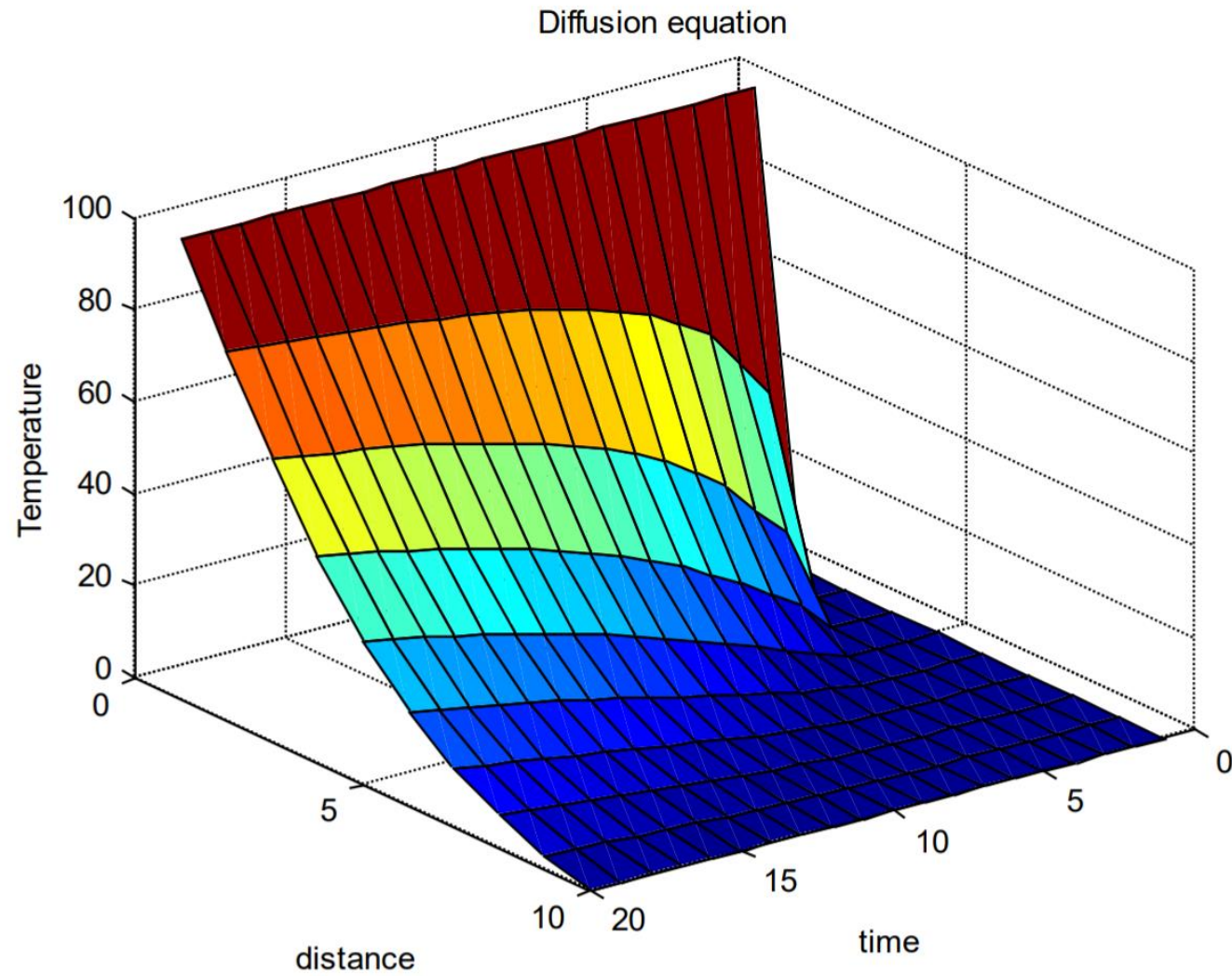
In this case, increasing time is horizontal, and the solution is beginning to approach the steady state at the right hand end.

Numerical Solution of the Diffusion Equation

Panopto bonus video

[illegible]

3D Plot



If a value of r greater than 0.5 is chosen, then the solution blows up and diverges!

A note on other solution methods

More robust, but more complex, methods (e.g. the Crank-Nicholson method) are described in the referenced text books.

In particular, the Crank-Nicholson method is unconditionally stable and has an order of convergence $O(h^2 + k^2)$

Part D – Numerical Solution of Hyperbolic Equations

Numerical methods for Hyperbolic PDEs

These can usually be solved by an **explicit numerical technique**, similar to that for parabolic equations. For example, take the one-dimensional wave equation,

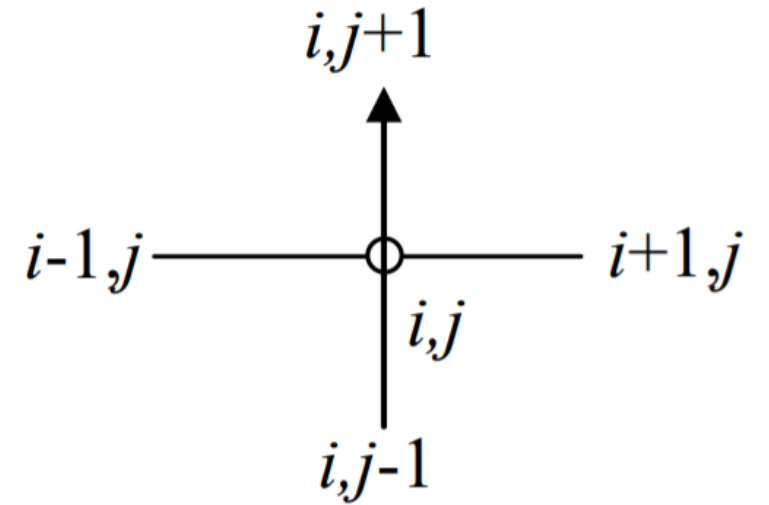
$$\frac{\partial^2 V}{\partial x^2} = \frac{\partial^2 V}{\partial t^2} \quad (\text{where we have set the phase velocity } c=1).$$

We can use a central difference formula to give:

$$\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{h^2} + O(h^2) = \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{k^2} + O(k^2)$$

We can simplify this if we use a square grid, $h = k$. In this case $V_{i,j}$ term cancels, and we get the recurrence relation:

$$V_{i,j+1} = V_{i-1,j} + V_{i+1,j} - V_{i,j-1} + O(h^4)$$



Error Analysis

As with parabolic equation solution, the method is conditionally stable i.e. will only converge if (Courant Number):

$$\frac{k}{h} \leq 1$$

(for the proof and discussion of the general case see, for example, Burden and Faires 8 th Edition, p. 718).

Solving initialisation problems

The method involves 3 time steps because $V_{i,j+1}$ depends on $V_{i,j-1}$ (two rows back in the scheme).

So we need **two** initial conditions, **the initial displacements $V_{i,1}$** & **the initial velocities, or gradients g_i** .

Using the forward difference formula for the first derivative,

$$g_{i,1} = \frac{V_{i,1} - V_{i,0}}{k}$$

We can set $V_{i,0} = V_{i,1} - k g_{i,1} = V_{i,1} - g_{i,1}$ for $k = 1$

Remember MATLAB index starts at 1 so 0 is a “false time row”!

We can then step through the remainder of the numerical solution as before using:

$$V_{i,j+1} = V_{i-1,j} + V_{i+1,j} + V_{i,j-1}$$

Numerical Solution of the Wave Equation

Panopto bonus video


B1 Numerical Algorithms

4 Lectures, MT 2022
Lecture four.

Bonus video 3 – Numerical Solution of the Wave Equation

Wes Armour

Errors/Typos/Questions to: wes.armour@city.ac.uk



1

★ 00:18

Numerical methods for Hyperbolic PDEs

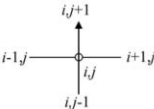
These can usually be solved by an **explicit numerical technique**, similar to that for parabolic equations. For example, take the one-dimensional wave equation,

$$\frac{\partial^2 v}{\partial x^2} = \frac{\partial^2 v}{\partial t^2} \quad (\text{where we have set the phase velocity } c=1).$$

We can use a central difference formula to give:

$$\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{h^2} + O(h^2) = \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{k^2} + O(k^2)$$

We can simplify this if we use a square grid, $h = k$. In this case $v_{i,j}$ term cancels, and we get the recurrence relation:

$$v_{i,j+1} = v_{i-1,j} + v_{i+1,j} - v_{i,j-1} + O(h^2)$$


2

★ 02:39

Error Analysis

As with parabolic equation solution, the method is conditionally stable i.e. will only converge if (Courant Number):

$$\frac{k}{h} \leq 1$$

(for the proof and discussion of the general case see, for example, Burden and Fairies 8 th Edition, p. 718).



3

★ 00:46

Solving initialisation problems

The method involves 3 time steps because $v_{i,j+1}$ depends on $v_{i,j-1}$ (two rows back in the scheme).

So we need **two** initial conditions, **the initial displacements $v_{i,1}$ & the initial velocities, or gradients $g_{i,1}$** .

Using the central difference formula for the first derivative,

$$g_{i,1} = \frac{v_{i,2} - v_{i,0}}{2k}$$

We can set $v_{i,0} = v_{i,2} - 2kg_{i,1} = v_{i,2} - 2g_{i,1}$ for $k = 1$

Remember MATLAB index starts at 1 so 0 is a "false time row"!

We can then step through the remainder of the numerical solution as before using:

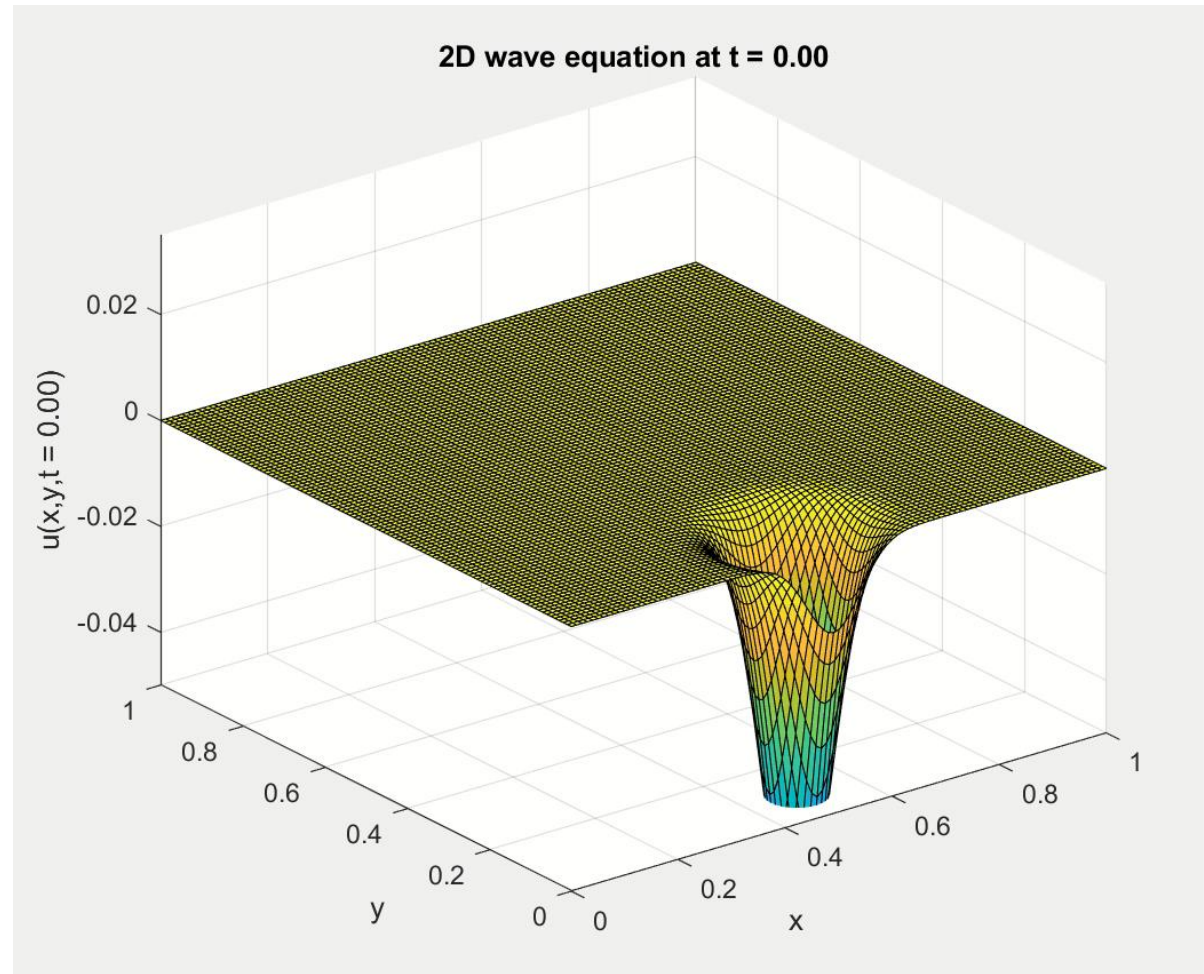
$$v_{i,j+1} = v_{i-1,j} + v_{i+1,j} + v_{i,j-1}$$


4

★ 03:34

Results

An example of the propagation of a wave formed by an object falling into a square pool of water. Starting from initial displacement at $t = 0$.



Results

Real world !!



Conclusions – what have we learnt?

- How to use finite difference methods to solve PDEs.
- How to use iterative solutions.
- Stability is an issue for explicit methods (parameters in our equations have constraints).
- Stability is guaranteed for implicit methods.
- The natural extension of the ideas described in this lecture is the Finite element method. This approach more easily handles boundary conditions involving derivatives and irregular shaped boundaries.



Summary of Lectures

You should now understand:

- Least squares fits and general polynomial regression (and limitations)
- Cross validation
- Different methods of numerical integration and how you calculate the associated errors
- Numerical errors associated with the finite precision of computers
- Numerical integration including the trapezium rule and Simpson's rule
- Numerical solutions of ODEs including Euler's Method, the predictor-corrector and Runge-Kutta methods.
- Methods for solving second-order ODEs
- Methods for solving the three different classes of PDEs.



That's all Folks!

Pease do respond to requests for feedback

this allows me to improve the lectures for
next years students (as last years students
feedback have improved the lectures for
you)