

Lecture 4:

PDE Boundary Value Problem: the Finite Difference Method

PDE Boundary Value Problem

- Need to solve a PDE in 2+ dimensions within a finite domain (volume) with given boundary conditions
- We will consider linear, elliptic PDEs
[e.g. $\nabla^2 \Phi = 0$; $\nabla^2 \Phi = \rho(\vec{r})$] and 2 classes of approach:
 - The Finite Difference Method (FDM)
 - The Finite Element Method (FEM)
- Will concentrate on
 - discretisation
 - reduction to a “Numerical Analysis” problem
 - how to deal with boundary conditions
(Dirichlet, Neumann, mixed)

For convenience we will stick to 2D ($\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ etc)

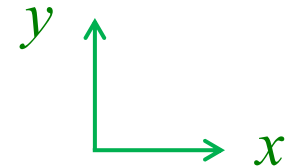
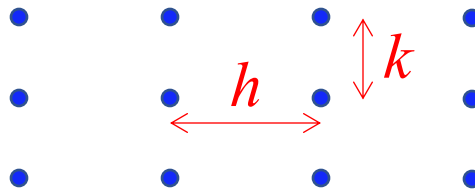
The Finite Difference Method (FDM)

- **Basic idea:** Discretise using grids...

$$\text{e.g. } \nabla^2 \Phi = \rho(\vec{r})$$

2d Cartesian:

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = \rho(x, y)$$



$$x_{i+1} = x_i + h \quad y_{j+1} = y_j + k$$

The Finite Difference Method (FDM)

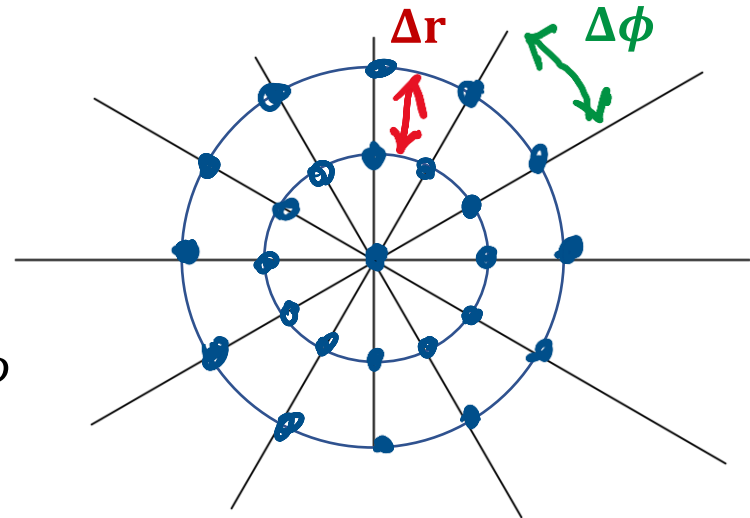
- **Basic idea:** Discretise using grids...

e.g. $\nabla^2 \Phi = \rho(\vec{r})$

Polar:

$$\frac{\partial^2 \Phi}{\partial r^2} + \frac{1}{r} \frac{\partial \Phi}{\partial r} + \frac{1}{r^2} \frac{\partial^2 \Phi}{\partial \phi^2} = \rho(r, \phi)$$

$$r_{i+1} = r_i + \Delta r \quad \phi_{j+1} = \phi_j + \Delta \phi$$



$$\nabla^2 \Phi = \rho$$

Functions

$$\rho(x, y) \rightarrow \rho(x_i, y_j) = \rho_{i,j} \quad \text{or} \quad \rho_I$$

$$\Phi(x, y) \rightarrow \Phi(x_i, y_j) = \Phi_{i,j} \quad \text{or} \quad \Phi_I$$

I is the “super-index”
running over all points

Note: **Indices** (*i, j*) are directly linked to (x,y) coordinates -> **convenient for interpretation**
“Super-index” *I* is convenient **for numerical implementation** (as we will see below)

Derivatives: apply CDA formulas for x- and y-derivatives, e.g.:

$$\frac{\partial^2 \Phi}{\partial x^2}(x, y) \rightarrow \frac{1}{h^2} [\Phi_{i+1,j} + \Phi_{i-1,j} - 2\Phi_{i,j}]$$

$$\frac{\partial^2 \Phi}{\partial y^2}(x, y) \rightarrow \frac{1}{k^2} [\Phi_{i,j+1} + \Phi_{i,j-1} - 2\Phi_{i,j}]$$

Note: for CDA a regular rectangular grid is required!

$$\nabla^2 \Phi = \rho$$

- **Easier to combine derivatives** if step size in x and y is the same, $h = k$:

$$\frac{\partial^2 \Phi}{\partial x^2}(x, y) \rightarrow \frac{1}{h^2} [\Phi_{i+1,j} + \Phi_{i-1,j} - 2\Phi_{i,j}]$$

$$\frac{\partial^2 \Phi}{\partial y^2}(x, y) \rightarrow \frac{1}{h^2} [\Phi_{i,j+1} + \Phi_{i,j-1} - 2\Phi_{i,j}]$$



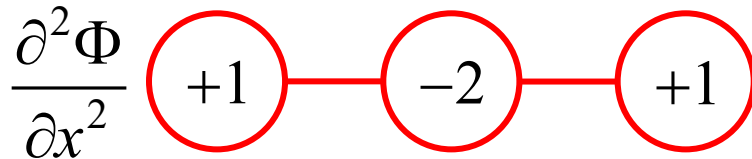
$$\nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} \rightarrow \frac{1}{h^2} [\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j}]$$

Note: sometimes, it is more convenient to use different step sizes in different directions

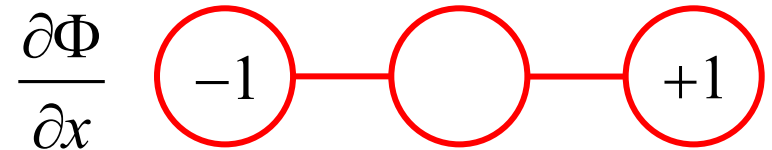
(e.g. heat distribution in a thin wire: transversal and longitudinal dimensions have different scales).

“Discretisation stencils” visualise these $O(h^2)$ difference approximations...

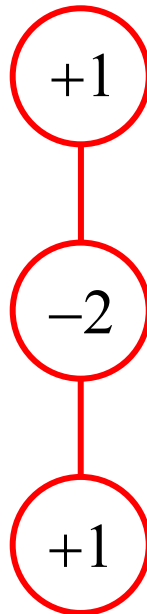
$$\frac{1}{h^2} [\Phi_{i+1,j} + \Phi_{i-1,j} - 2\Phi_{i,j}]$$



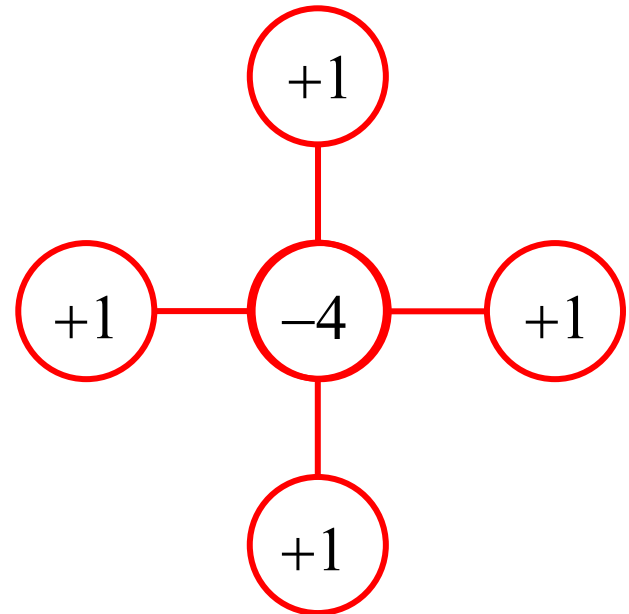
$$\frac{1}{2h} [\Phi_{i+1,j} - \Phi_{i-1,j}]$$



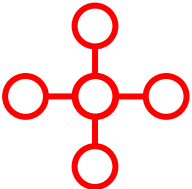
$$\frac{\partial^2 \Phi}{\partial y^2}$$



$$\nabla^2 \Phi$$



So, on a simple square grid $x_{i+1} = x_i + h$ $y_{j+1} = y_j + h$
 Poisson's equation $\nabla^2 \Phi = \rho$ in 2-d is approximated at the general point (i, j) by

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = h^2 \rho_{i,j} \quad (1)$$


Expect an equation like (1) at each grid point.

If there are N points at which $\Phi_{i,j}$ is unknown, we get N equations and N unknowns.

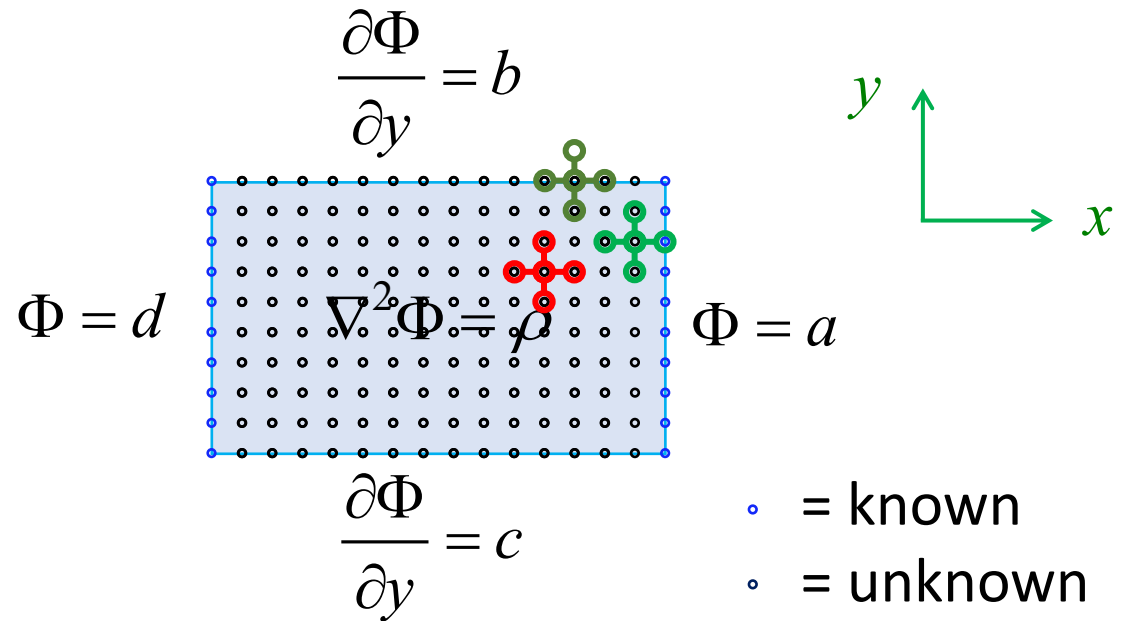
As PDE is linear, solve these as a matrix problem $\hat{M} \vec{\phi} = \vec{S} \quad (2)$

In super-index notation $\sum_J \hat{M}_{I,J} \phi_J = S_I$, \vec{S} contains values of ρ_I in the r.h.s. and known values of Φ from the boundary conditions

Note: double-indexing like $\phi_{i,j}$ is convenient for conceptualisation, but using super-index notation allows us to formulate the matrix problem in Eq. (2).

Implementation

Illustrate with
simple example:



$N = 14 \times 10 = 140$, so \hat{M} is 140×140

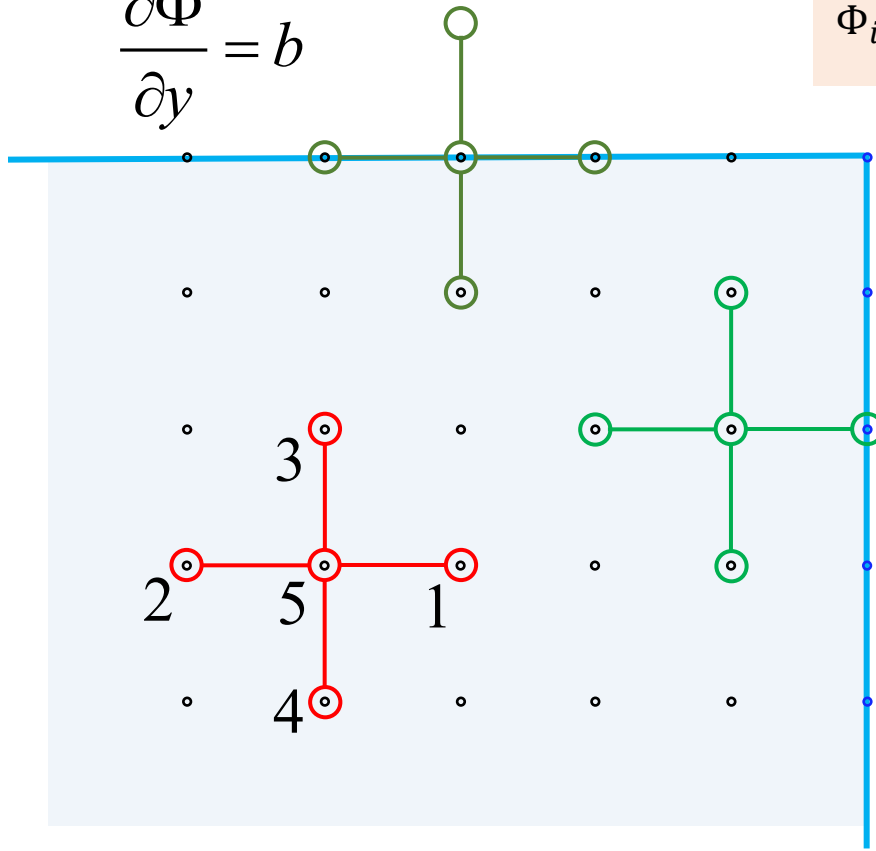
Superindices $I, J = 1, \dots, N$.

Consider equation ① at 3 representative points:

- A) in the interior of the region
- B) next to a Dirichlet boundary
- C) on a Neumann boundary

$$\frac{\partial \Phi}{\partial y} = b$$

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = h^2 \rho_{i,j} \quad (1)$$



$$\hat{M} \vec{\phi} = \vec{S}$$

$$\vec{\phi} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \vdots \\ \Phi_{140} \end{bmatrix}$$

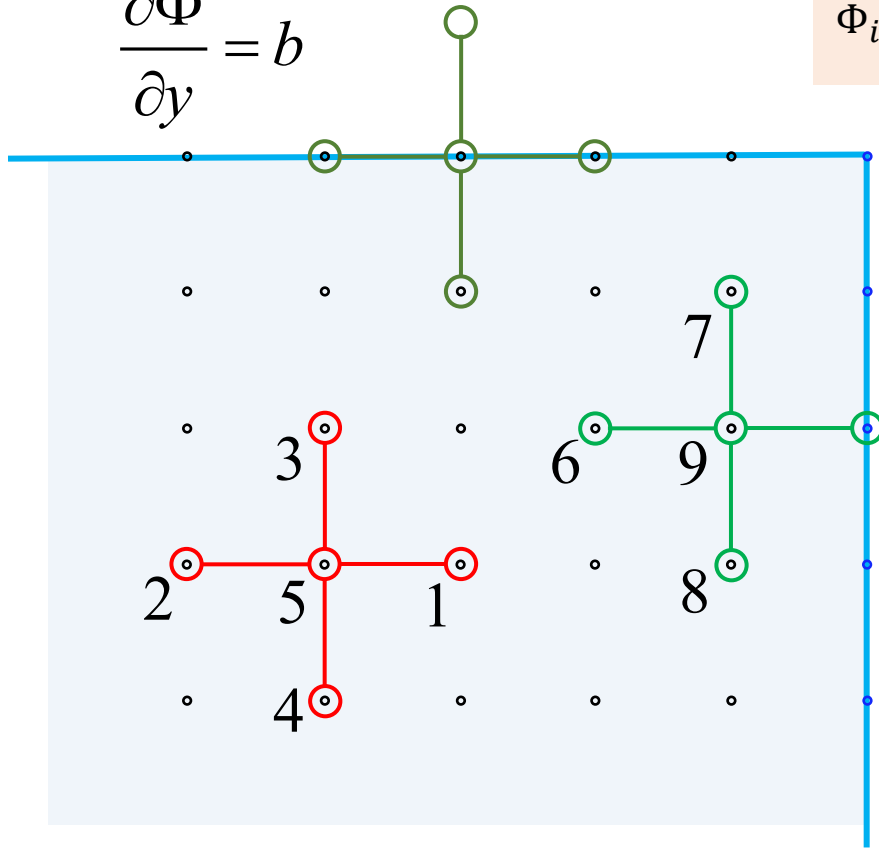
A) $I = 5; J = 1, 2, 3, 4, 5$. Eqn (1) $\rightarrow \Phi_1 + \Phi_2 + \Phi_3 + \Phi_4 - 4\Phi_5 = h^2 \rho_5$

So Row 5 of \hat{M} : $[1 \quad 1 \quad 1 \quad 1 \quad -4 \quad 0 \quad \dots]$; $S_5 = h^2 \rho_5$.

Note that \hat{M} is *sparse*: each row/column has max 5 non-zero elements.

$$\frac{\partial \Phi}{\partial y} = b$$

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = h^2 \rho_{i,j} \quad (1)$$



$$\hat{M} \vec{\phi} = \vec{S}$$

$$\vec{\phi} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \vdots \\ \Phi_{140} \end{bmatrix}$$

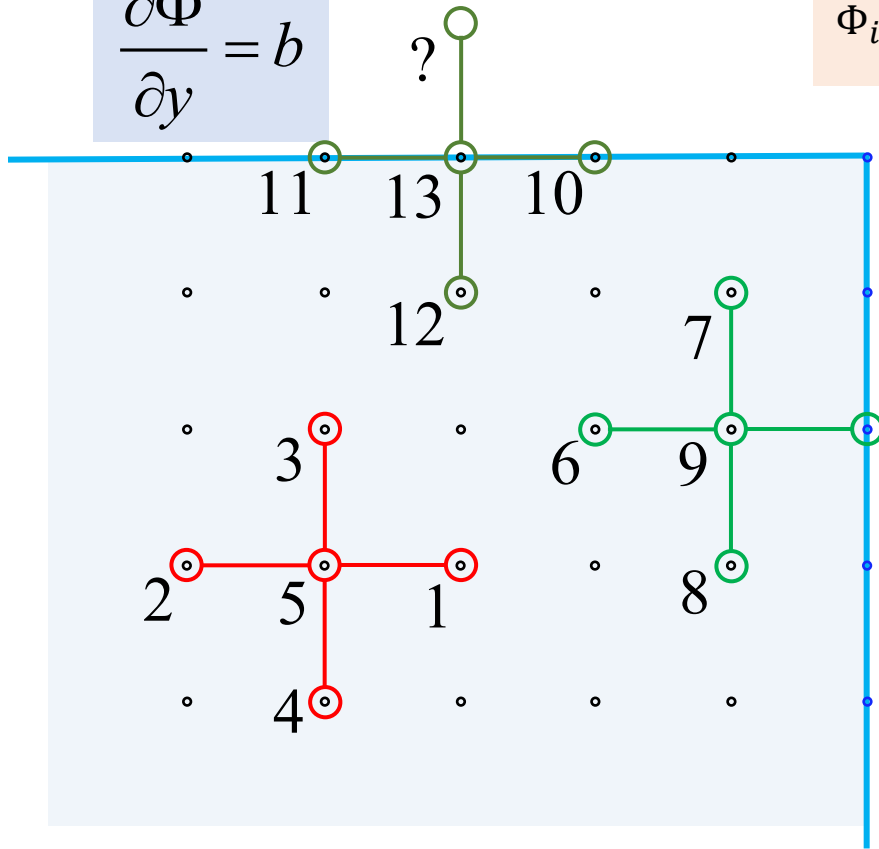
B) $I = 9; J = 6, 7, 8, 9.$ Eqn (1) $\rightarrow a + \Phi_6 + \Phi_7 + \Phi_8 - 4\Phi_9 = h^2 \rho_9$

So Row 9 of \hat{M} : $[\dots \underset{\substack{\uparrow \\ \text{column 5}}}{0} \quad 1 \quad 1 \quad 1 \quad -4 \quad \dots]; \quad S_9 = h^2 \rho_9 - a$

Note: If you do not implement any boundary condition, it is equivalent to setting $a = 0$.
I.e. the Dirichlet BC $\Phi = 0$ is the natural boundary condition for FDM method

$$\frac{\partial \Phi}{\partial y} = b$$

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = h^2 \rho_{i,j}$$



$$\hat{M} \vec{\phi} = \vec{S}$$

$$\vec{\phi} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \vdots \\ \Phi_{140} \end{bmatrix}$$

C) $I = 13$; $J = 10, 11, 12, 13, ?$. ① $\rightarrow \Phi_{10} + \Phi_{11} + \Phi_{?} + \Phi_{12} - 4\Phi_{13} = h^2 \rho_{13}$

For site “?” apply BC: $\frac{\partial \Phi}{\partial y} \approx \frac{1}{2h} (\Phi_{?} - \Phi_{12}) = b, \Rightarrow \Phi_{?} \approx 2hb + \Phi_{12}.$

So Row 13 of \hat{M} : $[\dots \underset{\substack{\uparrow \text{column 9}}}{0} \ 1 \ 1 \ 2 \ -4 \ \dots]$; $S_{13} = h^2 \rho_{13} - 2hb.$

So far, so good. But what if we have

1. A small defect in otherwise a homogeneous medium

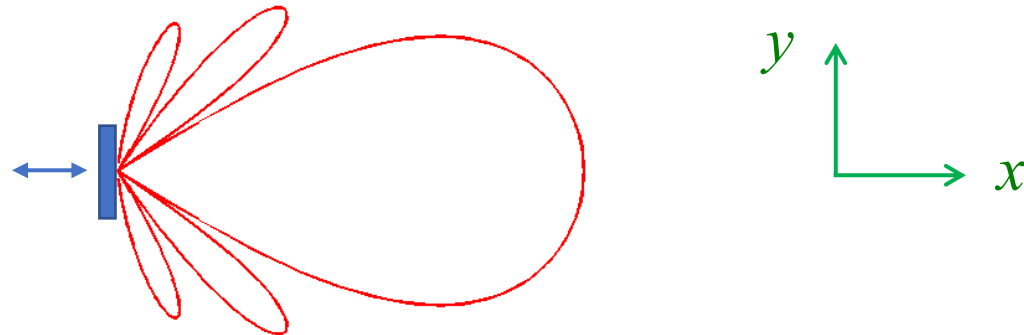
Need **more points to resolve the defect** BUT if grid is regular need **more points everywhere**. Will be computationally expensive!



2. Open regions – those with no obvious boundary?

Eg waves from a circular piston

Expect

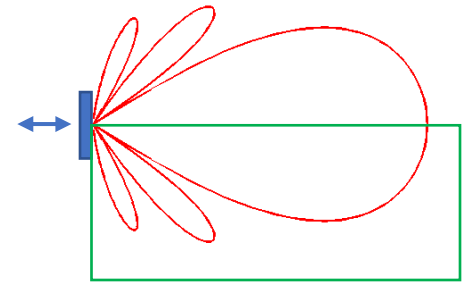


Radiation pattern is essentially 2D, and is, in principle, ∞ in extent.

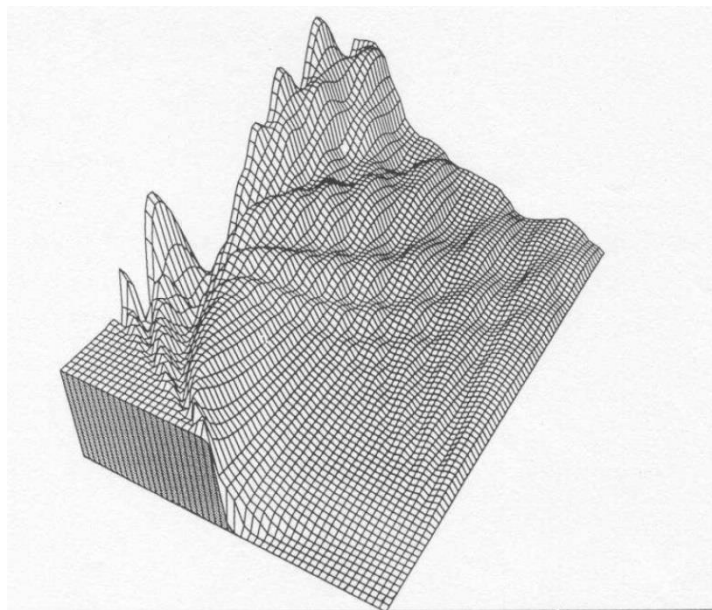
With a finite grid of points in x and y ,
what happens at (artificial) boundary?



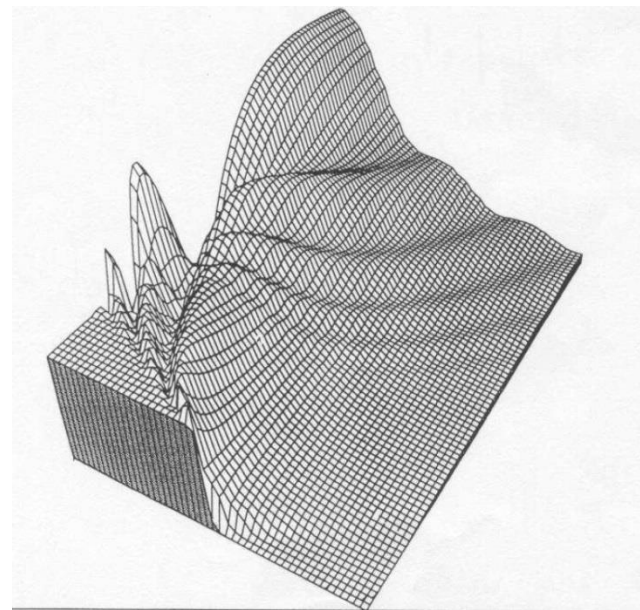
no grid point here: Numerically equivalent
to imposing the Dirichlet condition $\Phi = 0$



Leads to unwanted & unphysical reflections:



Bad



Better – how?

Solutions for open regions?

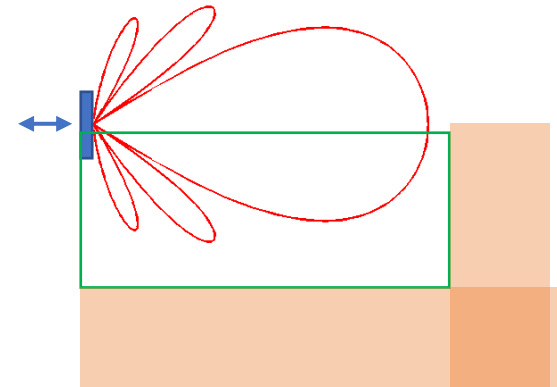
- Invent a pseudo-boundary condition to prevent reflections

If you just attempt to introduce an absorption along the boundary, it will give you a strong reflection anyway

Need to introduce artificial layers of an absorbing material, with a gradually increasing absorption away from the real boundaries

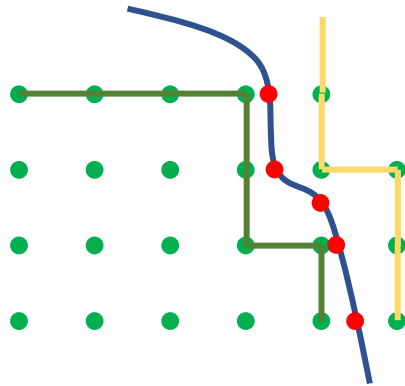
Keywords: “perfectly matched layer”,
“impedance matched”

It only works perfectly when you know exactly what kind of waves are “escaping” your window... In reality you can minimize reflection, but never fully suppress it.



- In practice, the only way to fully eliminate boundary reflections is.... to make the window size larger

Boundaries which don't follow the grid?

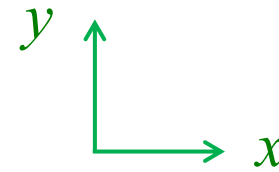
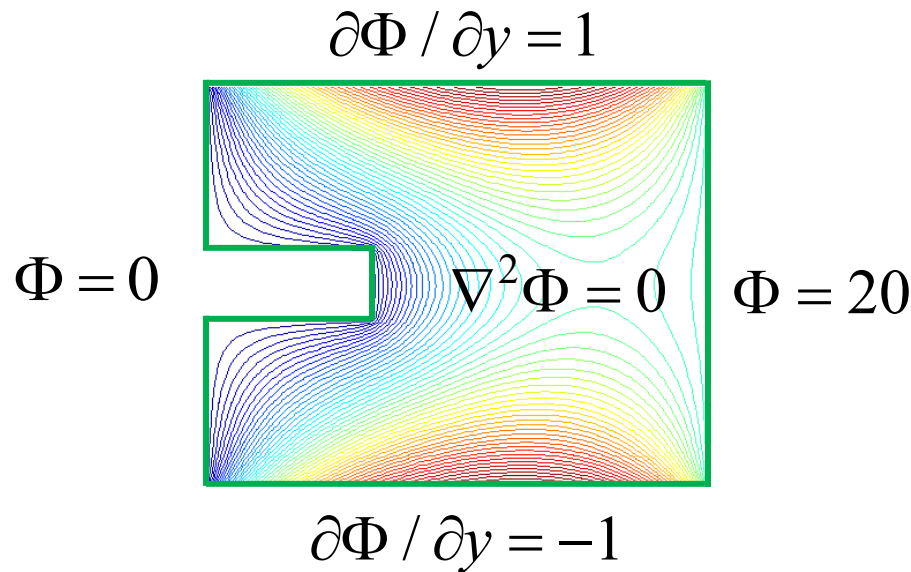


- more serious
- might *interpolate*: use • + •
usually $\rightarrow O(h)$ errors @ boundary
- might apply “*stair-step*” approximation –
average over 2 (false) boundaries .

Neither solution is particularly good...

Even worse, if $\frac{\partial \Phi}{\partial n}$ is specified (Neumann boundary), when the boundary is not \parallel to the grid

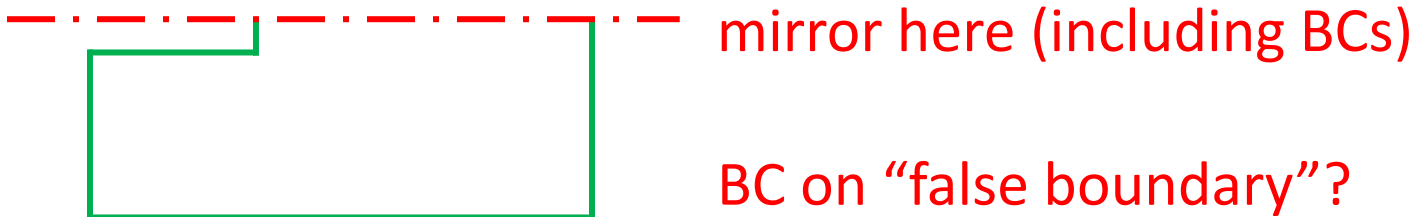
Using the symmetry



Easy geometry
Easy BCs

BUT $\hat{M}\vec{\phi} = \vec{S}$
is $N \times N$ problem,
 $\rightarrow T \sim N^3$

Use symmetry to reduce N & T :



Symmetry: Φ is the same just either side of mirror, so $\frac{\partial\Phi}{\partial y} = 0$

Gain? For same grid size, $N \rightarrow \sim N/2$, so $T \rightarrow \sim T/8$.

The FDM for eigenvalues & eigenvectors

PDE eigenvalue problem: $\hat{D}(\vec{r}, \nabla, \nabla^2)\Phi(\vec{r}) = \lambda\Phi(\vec{r})$

(e.g. electron wave-functions in a crystal, RF and optical waveguides)

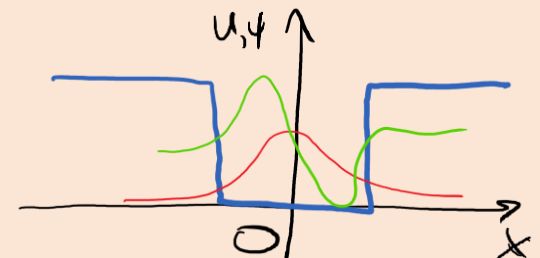
Discretise derivatives on a grid: $\hat{M}\vec{\Phi} = \lambda\vec{\Phi}$

- obtain a matrix eigenvalue problem
- solve numerically for (approx) eigenvalues λ
eigenvectors $\vec{\Phi}$

BUT: careful with symmetries!

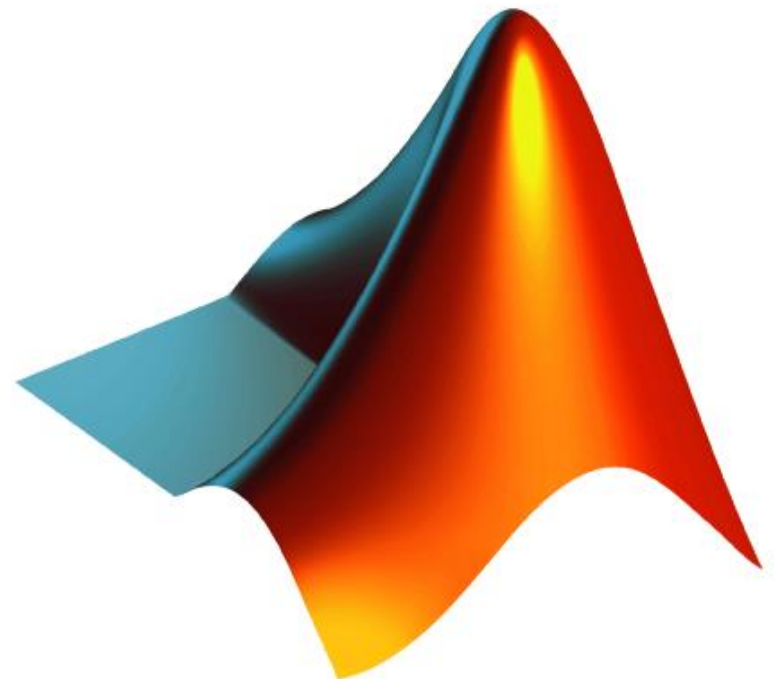
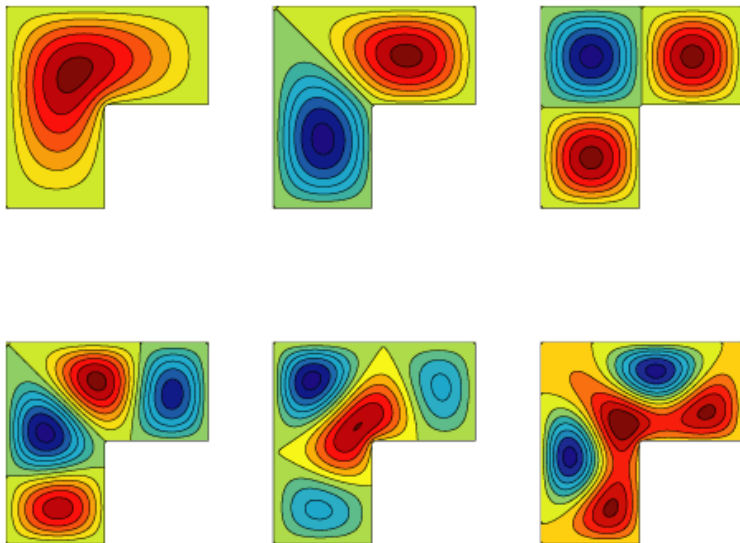
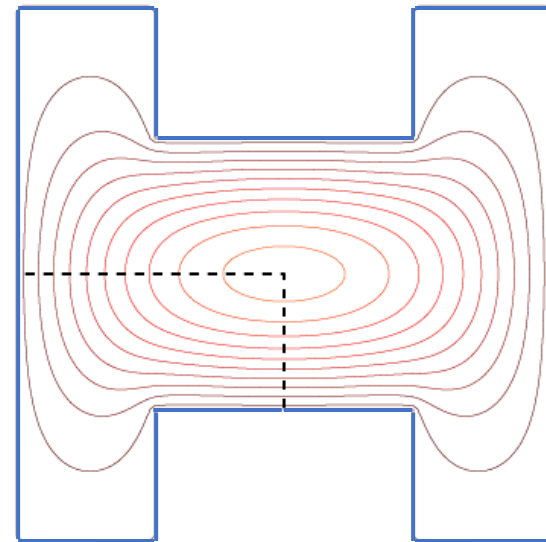
E.g. a symmetric quantum well has even and odd states

=> Need to run simulation twice with different BCs at $x = 0$.



Example

An RF
waveguide



How this topic will be assessed:

- Typical questions on exam:

Discretise a given 2D Boundary Value Problem PDE using Finite Difference Method:

- *Write down the resulting equation for a generic point away from any boundaries;*
- *Write down the equation for a point a particular boundary: show how to implement a particular type of BC.*

Discuss issues with implementing FDM in complex geometries

See further examples in worksheet 2

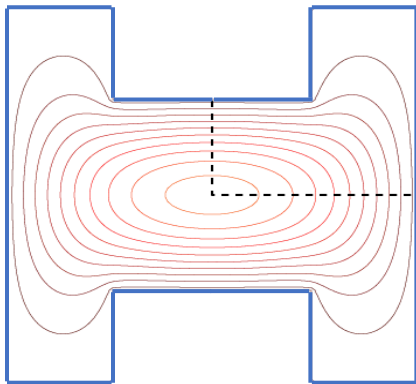
Lecture 5:

The Finite Element Method

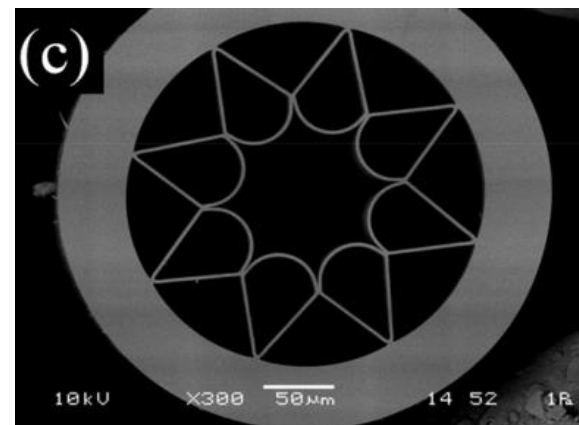
Difficulties with FDM

- The Finite Difference Method is easy to implement, but it has two weaknesses:
 - ***It requires a regular grid*** => hence computationally challenging for problems where a mixture of large-scale and small-scale features is present
 - ***Difficult to implement boundary conditions*** for boundaries which are not parallel to the grid

A straightforward application of FDM



A challenge for FDM

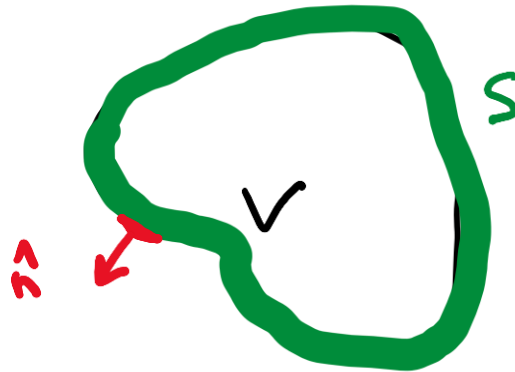


The Finite Element Method (FEM)

- FEM takes a different approach to solve a BVP problem, by re-phrasing it in the so-called “weak form”.
 - In the *standard (strong) formulation* the conditions are imposed on the solution directly by requiring that it must satisfy:
 - 1) the PDE (in each and every point of space inside the domain);
 - 2) the boundary conditions.
 - In the *weak formulation* the solution is not required to satisfy the equation pointwise. Instead, it must satisfy an integral identity involving a set of test functions.

An example (Poisson's equation)

The equation: $\nabla^2 \Phi = \rho$, $(x, y) \in V$ ← the domain
(e.g. for a rectangle $x \in [0, L_x], y \in [0, L_y]$)




BCs: $\Phi = A$, $(x, y) \in S_1$ ← Dirichlet BCs along some sections of the domain boundary S

$\hat{n} \cdot \nabla \Phi = 0$, $(x, y) \in S_2$ ← Neumann BCs along the rest of the domain boundary S

Note: other BCs are also possible, but we will focus on these two types as the most common.

- Multiply both parts by a “test” function η and integrate over the entire domain:

$$\int_V \eta \nabla^2 \Phi \, dx dy = \int_V \rho \eta \, dx dy$$


What is this test function?

We will require that:

- 1) η is defined (finite) in the entire domain V ;
- 2) $\eta = 0$ for $(x, y) \in S_1$ - i.e. along all the parts of the boundary with Dirichlet BCs

- Let's re-write the integral in the left-hand side in a more symmetrical form:

$$\begin{aligned} \int_V \eta \nabla^2 \Phi \, dx dy &= [\text{Green's identity}] \\ &= \underbrace{\oint_S \eta (\hat{n} \cdot \nabla \Phi) dS}_{=0 \text{ because:}} - \int_V (\nabla \Phi \cdot \nabla \eta) \, dx dy \end{aligned}$$

- =0 because:**
- a) $\eta = 0$ for $(x, y) \in S_1$
 - b) $\hat{n} \cdot \nabla \Phi = 0$ for $(x, y) \in S_2$

$$\begin{aligned} \nabla^2 \Phi &= \rho, & (x, y) \in V \\ \Phi &= A, & (x, y) \in S_1 \\ \hat{n} \cdot \nabla \Phi &= 0, & (x, y) \in S_2 \end{aligned}$$

$$\begin{aligned}\nabla^2 \Phi &= \rho, & (x, y) \in V \\ \Phi &= A, & (x, y) \in S_1 \\ \hat{n} \cdot \nabla \Phi &= 0, & (x, y) \in S_2\end{aligned}$$

- Hence we obtain:

$$-\int_V (\nabla \Phi \cdot \nabla \eta) dx dy = \int_V \rho \eta dx dy$$

This is the **weak formulation** of our BVP problem

Note: It is easy to see that a solution of our initial BVP problem is automatically a solution of the weak problem (we just derived it!). The opposite, however, is not so obvious: if I find a function Φ which satisfies the above integral condition for any test function η , would this function be a solution of the initial BVP problem? The answer is yes. And you can prove it by using variational calculus. But we are not going to do it here.

- We can re-write it in a shorter notation:

$$\langle \nabla \Phi, \nabla \eta \rangle = -\langle \rho, \eta \rangle$$

$$\text{where } \langle f, g \rangle = \int_V (f g) dx dy$$

Bilinear form $L(\Phi, \eta)$
(i.e. it is linear in each of its two arguments)

Linear form $F(\eta)$

Note 2: $L(A, B) = L(B, A)$ – this is going to be useful in the numerical implementation!

Note 1: Φ and η are unknown functions, but ρ is the right-hand side function given in the problem (hence it is not treated as an argument in this linear form)

Discretisation of the weak formulation: Galerkin method

$$\langle \nabla \Phi, \nabla \eta \rangle = -\langle \rho, \eta \rangle \quad \text{where } \langle f, g \rangle = \int_V (f g) dx dy$$

- Instead of an infinite space of continuous functions in V , we will restrict our search by a discrete sub-space of such functions.
- This subset is defined by a finite set of functions, the **basis**: $v_k(x, y)$, $k = 1, 2, \dots, N$
 - We will restrict our search by functions Φ which can be represented as a linear combination of the basis functions:

$$\Phi(x, y) \approx \Phi_{FEM}(x, y) = \sum_{k=1}^N \phi_k v_k(x, y)$$

Note: at this stage we introduce a discretisation error!

- Use the basis functions as test functions (N test functions in total $\Rightarrow N$ equations):

$$\sum_{k=1}^N \phi_k \langle \nabla v_k, \nabla v_j \rangle = -\langle \rho, v_j \rangle, \quad j = 1, 2, \dots, N$$

OR we can write this as:

$$\hat{L} \vec{\phi} = \vec{F}$$

- Matrix problem for N unknown coefficients ϕ_k

where $L_{k,j} = L(v_k, v_j) = \langle \nabla v_k, \nabla v_j \rangle$, $F_j = F(v_j) = -\langle \rho, v_j \rangle$

Discretisation of the weak formulation: Galerkin method

$$\Phi(x, y) \approx \Phi_{FEM}(x, y) = \sum_{k=1}^N \phi_k v_k(x, y)$$

$$\hat{L}\vec{\phi} = \vec{F}$$

$$L_{k,j} = L(v_k, v_j) = \langle \nabla v_k, \nabla v_j \rangle,$$
$$F_j = F(v_j) = -\langle \rho, v_j \rangle$$



Boris Galerkin
1871-1945

https://en.wikipedia.org/wiki/Boris_Galerkin

- This discretisation procedure is known as Galerkin method
- There are many choices of basis functions – it is a family of methods!
- The **discretisation error** is defined by:
 - Our choice of the basis functions (i.e. how well the actual solution can be approximated by a linear combination of the basis function);
 - The size of the basis (i.e. the number of the basis functions).

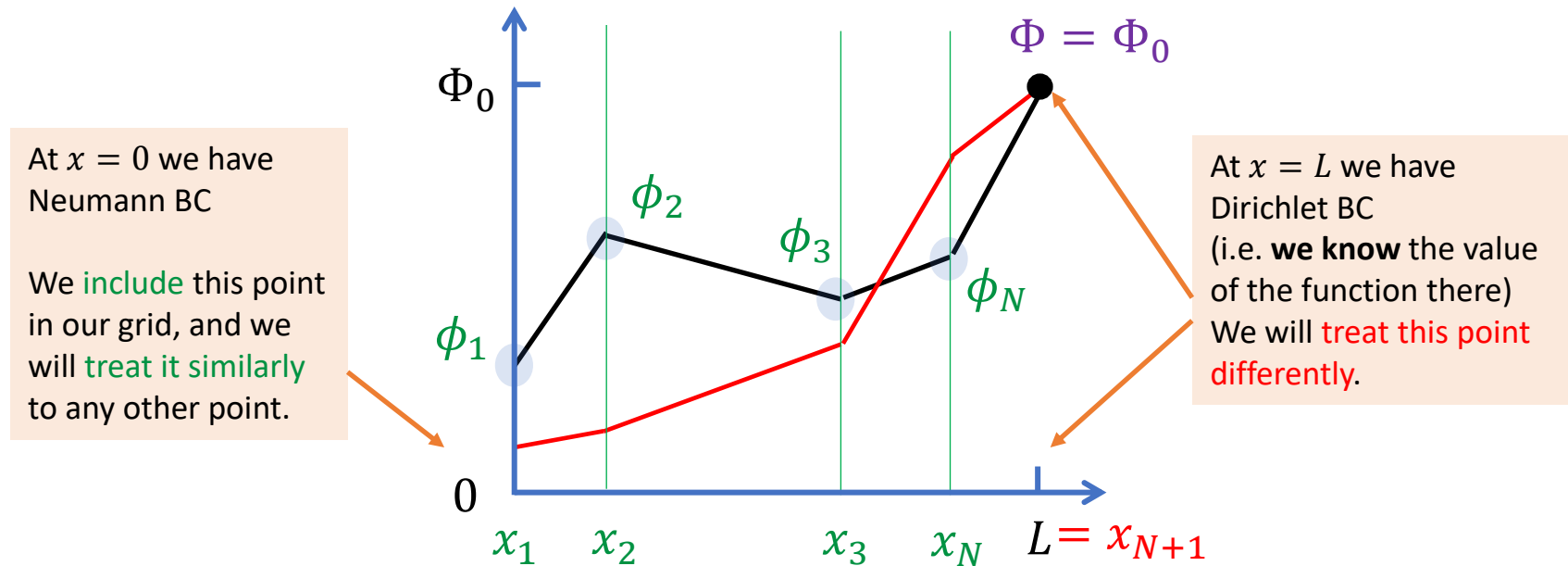
Note: these two statements are linked together (e.g. the more basis functions you choose, the better you can approximate the solution?), but they are not necessarily the same thing! For example, if you are trying to approximate a non-periodic solution with a set of periodic functions (such as Fourier series), you will introduce an error no matter how many functions you take in the expansion.

(unlike in the Finite Difference method) **The error is difficult to quantify** since you do not know the true solution!

But you can try e.g. changing the basis and/or increasing the number of functions, to see if it introduces any noticeable changes to your solution.

- Piecewise-linear(/polynomial) functions is the most popular choice in FEM solvers

A 1D example: $\frac{d^2\Phi}{dx^2} = \rho, \quad x \in [0, L], \quad \frac{d\Phi}{dx}(0) = 0, \Phi(L) = \Phi_0$

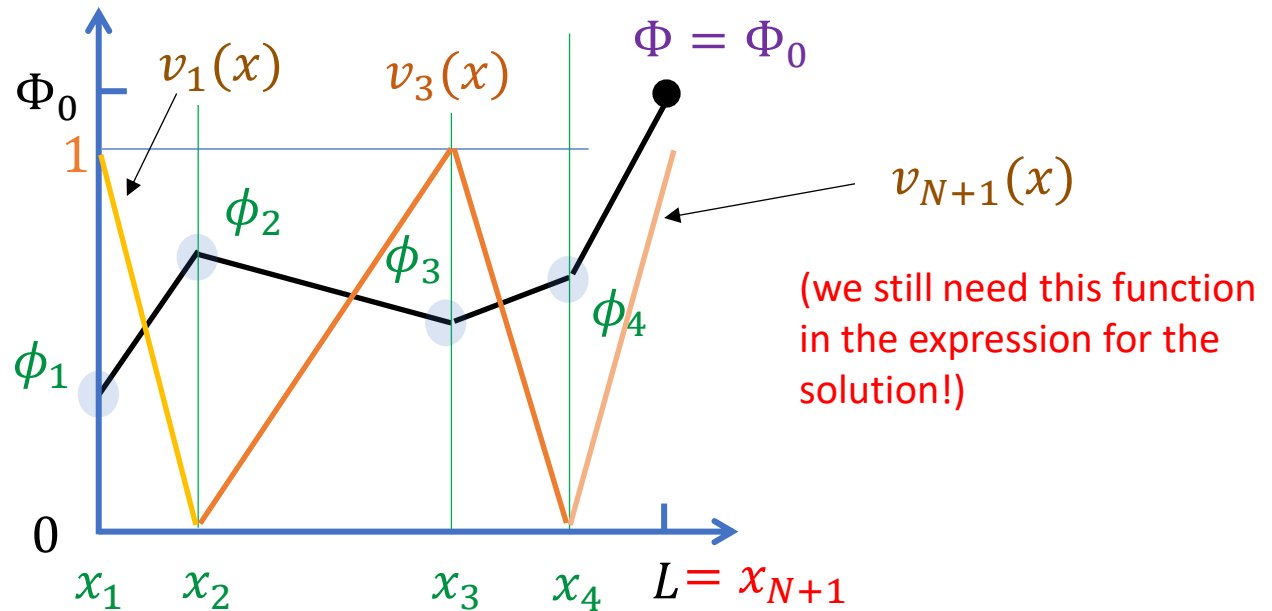


- Choose positions of N nodes x_j - **no need to space them equally!**
- Consider piecewise-linear functions
- The FEM solution is defined by the values ϕ_j at the nodal points.

BASIS set functions $v_k(x)$?

BASIS set: “tent” functions

$$v_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}}, & x \in [x_{k-1}, x_k] \\ \frac{x_{k+1} - x}{x_{k+1} - x_k}, & x \in [x_k, x_{k+1}] \\ 0, & \text{otherwise} \end{cases}$$



$$\Phi(x) \approx \Phi_{FEM}(x) = \sum_{k=1}^N \phi_k v_k(x) + \Phi_0 v_{N+1}(x)$$

Note: this basis is not orthogonal!

Indeed: $\langle v_k, v_j \rangle = \int_0^L v_k v_j dx = 0$ if $|j - k| \geq 2$, but $\langle v_k, v_{k\pm 1} \rangle \neq 0$!

“Tent” functions: calculate matrix elements and rhs vector

$$\hat{L}\vec{\phi} = \vec{F}$$

$$L_{k,j} = L(v_k, v_j) = \left\langle \frac{dv_k}{dx}, \frac{dv_j}{dx} \right\rangle,$$

$$F_j = F(v_j) = -\left\langle \rho, v_j \right\rangle$$

$$\langle f, g \rangle = \int_0^L (fg) dx$$

- Using the analytical formulas for the “tent” functions, can calculate the integrals
(do it yourself as a useful exercise!)

$$\left\langle \frac{dv_k}{dx}, \frac{dv_j}{dx} \right\rangle = \begin{cases} \frac{x_{k+1} - x_{k-1}}{(x_k - x_{k-1})(x_k - x_{k+1})}, & \text{if } j = k, k \neq 1 \\ \frac{1}{x_2 - x_1}, & \text{if } j = k = 1 \\ \frac{-1}{x_{k+1} - x_k}, & \text{if } j = k + 1 \\ 0, & \text{if } j \neq k \text{ or } j \neq k \pm 1 \end{cases}$$

Note 1: because the bilinear form is symmetric, $L_{k,j} = L_{j,k}$

Note 2: for the “tent” functions, the resulting matrix L **will contain many zero elements**



1+2 => \hat{L} is a **symmetric sparse** matrix. There are special numerical algorithms for solving such matrix problems efficiently!

The rhs vector:

$$-\langle \rho, v_j \rangle = -\int_0^L \rho(x) v_j(x) dx = F_j$$

- For a given rhs function $\rho(x)$ need to calculate these integrals either analytically or numerically (*e.g. using trapezoidal rule*)

E.g. for $\rho(x) = \rho_0 = \text{const.}$ easy to obtain $F_j = -\rho_0 \int_0^L v_j(x) dx = \begin{cases} -0.5\rho_0(x_{j+1} - x_{j-1}), & j = 2, 3, \dots, N \\ -0.5\rho_0(x_2 - x_1), & j = 1 \end{cases}$

Treating the boundary conditions

$$\Phi(x) \approx \Phi_{FEM}(x) = \sum_{k=1}^N \phi_k v_k(x) + \Phi_0 v_{N+1}(x)$$

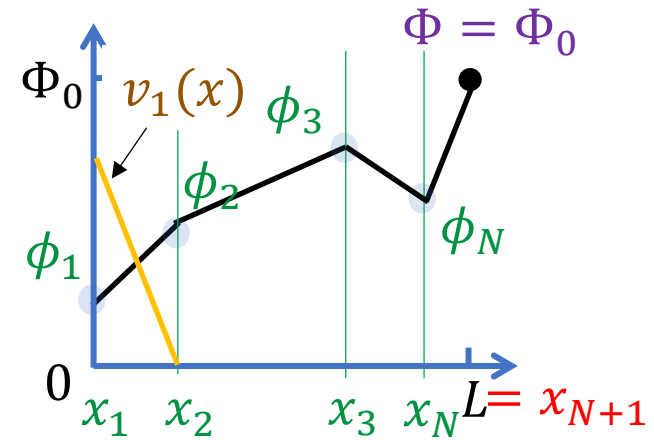
- Neumann BC: $\frac{d\Phi}{dx}(0) = 0$

(and more generally, in 2D and 3D: $\hat{n} \cdot \nabla \Phi = 0$)

- ...is taken into account by simply including this point ($x_1 = 0$) in the grid!
- It is treated in the same way as all other points

Note: the shape of $v_1(x)$ (“half-tent”) is qualitatively different from all other “tent” functions. But all formal expressions for $L(v_k, v_j)$ and $F(v_k)$ never assume similar or identical shapes of basis functions. No adjustments/modifications are needed.

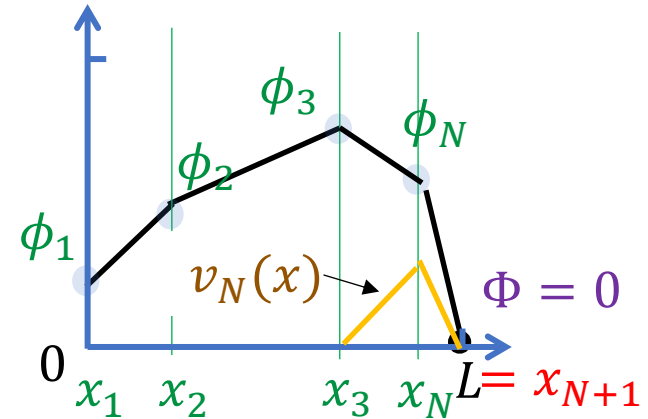
- This is the **natural boundary condition** for FEM



Note 2: a non-zero flux at the boundary, $\hat{n} \cdot \nabla \Phi = A \neq 0$, would **require an adjustment to the weak formulation**. There are well-established procedures, but this is beyond the scope of this module.

Treating the boundary conditions

$$\Phi(x) \approx \Phi_{FEM}(x) = \sum_{k=1}^N \phi_k v_k(x) + \cancel{\Phi_0 v_{N+1}(x)}$$

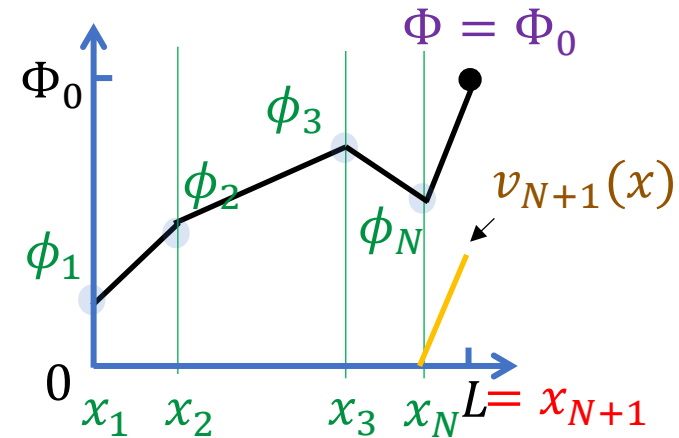


- Zero Dirichlet BC: $\Phi(L) = 0$ (*i.e.* $\Phi_0 = 0$)
 - The end point $x = L$ is **not included in the grid**
 - The FEM solution is still represented as a linear combination of N “tent” functions located at N grid points
 - The N th “tent” function, $v_N(x)$, formally **extends beyond the grid** and includes the end point $x_{N+1} = L$.
 - This extension must be taken into account when calculating the matrix element $L_{N,N} = L(v_N, v_N)$, and the rhs vector element $F_N = F(v_N)$.
(i.e. you will need to include the interval $[x_N, L]$ in the corresponding integrals)
 - This is **NOT** a natural boundary condition, but still requires minimal effort

Treating the boundary conditions

$$\Phi(x) \approx \Phi_{FEM}(x) = \sum_{k=1}^N \phi_k v_k(x) + \Phi_0 v_{N+1}(x)$$

- Non-zero Dirichlet BC: $\Phi(L) = \Phi_0 \neq 0$



- The end point $x = L$ is **not included in the grid**
- The N th “tent” function, $v_N(x)$, formally **extends beyond the grid** and includes the end point $x_{N+1} = L$.
- The FEM solution is represented as a linear combination of **$N+1$** “tent” functions located at N grid points **+ the end point $x_{N+1} = L$ (“half-tent”)**
- When constructing the matrix equation, you still have N unknowns (ϕ_k), and you are still using N test functions $v_k(x)$, $k = 1, 2, \dots, N$. But you need to adjust the “general” formula to take into account the extra term (v_{N+1} function) in the expansion for Φ .

For that, let's rewind a few slides back to when we introduced discretization of the weak form

(rewind few slides back)

Discretisation of the weak formulation: Galerkin method

$$\langle \nabla \Phi, \nabla \eta \rangle = -\langle \rho, \eta \rangle \quad \text{where } \langle f, g \rangle = \int_V (f g) dx dy$$

- Instead of an infinite space of continuous functions in V , we will restrict our search by a discrete sub-space of such functions.
- This subset is defined by a finite set of functions, the **basis**: $v_k(x, y)$, $k = 1, 2, \dots, N$
 - We will restrict our search by functions Φ which can be represented as a linear combination of the basis functions **and the additional function v_{N+1}** :

$$\Phi(x, y) \approx \Phi_{FEM}(x, y) = \sum_{k=1}^N \phi_k v_k(x, y) + \Phi_0 v_{N+1}(x)$$

- Use the basis functions as test functions (N test functions in total $\Rightarrow N$ equations):

$$\sum_{k=1}^N \phi_k \langle \nabla v_k, \nabla v_j \rangle + \Phi_0 \langle \nabla v_{N+1}, \nabla v_j \rangle = -\langle \rho, v_j \rangle, \quad j = 1, 2, \dots, N$$

OR we can write this as:

$$\hat{L} \vec{\phi} = \vec{F}$$

- Matrix problem for N unknown coefficients ϕ_k

$$\text{where } L_{k,j} = L(v_k, v_j) = \langle \nabla v_k, \nabla v_j \rangle, \quad F_j = -\langle \rho, v_j \rangle - \Phi_0 \langle \nabla v_{N+1}, \nabla v_j \rangle$$

FEM in 2D and 3D

- The main applications of FEM are in 2D and higher dimensions

(this is when we face main issues with finite difference method)

- The process is very similar

E.g. Poisson's equation in 2D domain with zero Dirichlet BC, $\Phi = 0$, along the circumference

- The most popular meshes are triangular (for 2D) and tetrahedral (for 3D)

- Select nodes inside the domain – they will correspond to the unknowns ϕ_k

- Select nodes along the boundary.

For Dirichlet BC, there are no unknowns associated with these nodes, but they are needed for meshing.

- Connect the nodes and create the mesh.

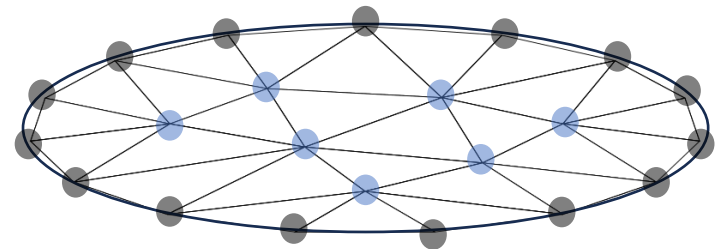
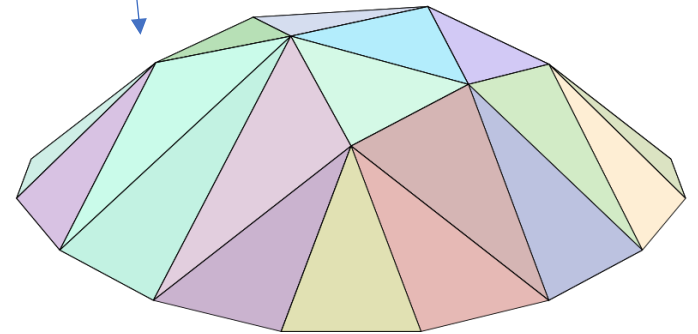
- Use the 2D analogue of “tent” functions. Construct the matrix equation.

Compared to 1D, there will be more overlaps with the neighbouring “tent” functions. But the functions are still compact, and so the matrix will still be sparse (and symmetric).

=> Very useful for efficient solvers, especially for eigenvalue problems!

An example of a piecewise-linear FEM solution in 2D geometry

Picture by Oleg Alexandrov - self-made, with en:MATLAB., <https://commons.wikimedia.org/w/index.php?curid=2263047>



FEM vs FDM

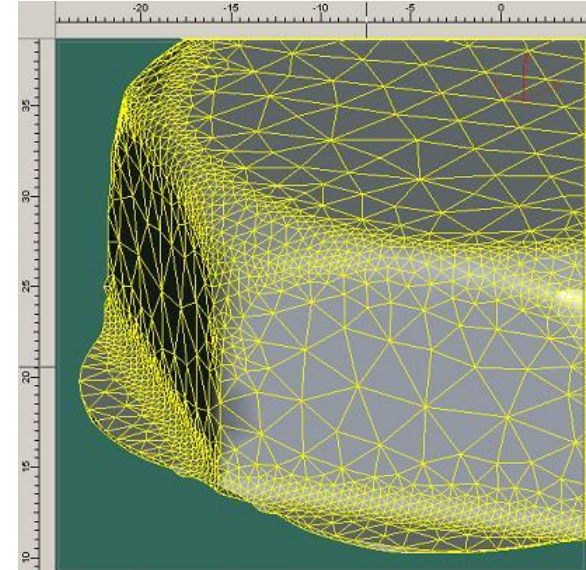
- The Finite Element Method (FEM) requires some effort to implement. But the benefits are:

1) Advanced options for mesh: no specific requirements/constraints as for the size or shape of individual elements

2) The meshing is better suited for complex shapes

3) Much easier handling of Neumann boundary conditions

Because of this, FEM is often the default method for complex 2D and 3D geometries.



- Many physical problems are eventually reduced to a few well-known equations: for many applications you can use available FEM solvers (*but most of them are not free!*)

Note: despite the similar names, Finite VOLUME Method (FVM) and Finite ELEMENT Method (FEM) are **two very different methods!**

The Finite Volume Method is very common in fluid dynamics/astrophysics. It is beyond the scope of this course. (*But you may learn about it later if you are taking Comp Astro*).

How this topic will be assessed:

- Typical questions on exam:

Discuss advantages/disadvantages of FEM compared to FDM

For a given 1D Boundary Value Problem:

- *Derive the weak formulation;*
- *Discretise using 2-3 points and linear “tent” functions as the basis, derive the resulting matrix equations;*
- *Implement Dirichlet and Neumann boundary conditions.*

See further examples in worksheet 2

Lecture 6:

Initial Value Problem parabolic PDEs
finite difference methods

General problem: a PDE with a combination of time and space derivatives, e.g.

$$i\hbar \frac{\partial}{\partial t} \Psi(\vec{r}, t) = \left(-\frac{\hbar^2}{2m} \nabla^2 + V(\vec{r}) \right) \Psi(\vec{r}, t) \quad \text{- Schrödinger equation (parabolic)}$$

$$\frac{\partial^2}{\partial t^2} u(x, t) = c^2 \frac{\partial^2}{\partial x^2} u(x, t) + f(x, t) \quad \text{- wave equation (hyperbolic)}$$

- Auxiliary conditions: combination of **IVP** (time) and **BVP** (space):

$$\begin{aligned} \text{e.g.} \quad u(x, t = 0) &= U_0(x), & u(0, t) &= 0, \\ \frac{\partial}{\partial t} u(x, t = 0) &= V_0(x), & u(L, t) &= 0. \end{aligned}$$

- Time and space are treated differently: such systems are often referred to as (1+N)-dimensional problems (e.g. (1+3)D Schrödinger equation)
- We will focus on (1+1)D linear parabolic problems

$$\text{e.g.} \quad i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

Semi-Analytical method

$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

- Eigenstates form a basis set for expansion (similar to Fourier transforms, but more generic):

$$u(x, t) = e_{\omega}(x) \exp(-i\omega t) \quad \text{- Stationary analysis}$$

$$\omega \cdot e_{\omega}(x) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) e_{\omega}(x) \quad \text{- Solve as an eigenvalue problem (for a given potential, solve e.g. numerically by discretising } x \text{ – see Lectures 2 and 3)}$$

$$u(x, t = 0) = U_0(x) = \int C_{\omega} e_{\omega}(x) d\omega \quad \text{- Expand the initial solution into the basis set}$$

$$C_{\omega} = \int u(x) e_{\omega}^*(x) dx \quad \text{(in numerics, all integrals will be replaced by the sums over } N \text{ normalised eigen-states)}$$

- Hence at any time t the solution is given by:

$$u(x, t) = \int C_{\omega} e_{\omega}(x) e^{-i\omega t} d\omega$$

Semi-Analytical method

$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

- Numerical **errors only due to discretisation of x** (i.e. using finite basis set), but there is no discretisation of time
- Works only for **LINEAR** equations
- The method is **computationally expensive**:

$$\omega \cdot e_{\omega}(x) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) e_{\omega}(x)$$

- Full eigenvalue problem (i.e. need to find ALL eigenvectors and eigenvalues).
Computational time scales as $O(N^3)$

$$C_{\omega} = \int u(x) e_{\omega}^*(x) dx$$

- Compute all expansion coefficients:
computational time $\sim O(N^2)$

$$u(x, t) = \int C_{\omega} e_{\omega}(x) e^{-i\omega t} d\omega$$

- Resulting solution as a sum over all eigenstates: computational time $\sim O(N^2)$

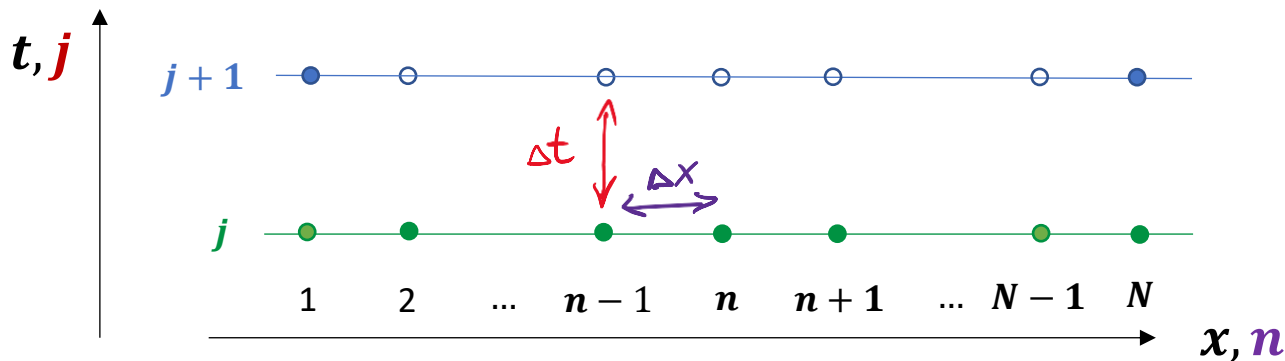
- **An elegant but slow method**

Finite-Difference methods: general idea

- Discretize time and space coordinates:

$x \rightarrow x_n = n \cdot \Delta x$ - For **all spatial coordinates** we normally use **regular grid**
(such that CDA for all spatial derivatives can be applied)

$t_{j+1} = t_j + \Delta t$ - For **time** we can use schemes with **adaptive step**.
Unless the regular grid is required by a specific method (e.g. Leapfrog), we shall assume that Δt can vary with each iteration



○ need to obtain

● fixed by BVP

● (for $j = 1$): fixed by IVP
(for $j > 1$): obtained at the previous iteration

Forward-Time Centred-Space method (FTCS)

$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

- Discretise time and space. Use FDA for time derivative, CDA for space:

$$t = j\Delta t, j = 0, 1, 2, \dots$$

$$x = n\Delta x, n = 0, 1, 2, \dots$$

(e.g. solving on $t \in [0, T]$,
 $x \in [0, L]$ intervals)

$$u(\mathbf{x}_n, \mathbf{t}_j) = u_n^{(j)}$$

time index
space index

$$\Rightarrow i \frac{1}{\Delta t} [u_n^{(j+1)} - u_n^{(j)}] = \frac{1}{(\Delta x)^2} (2u_n^{(j)} - u_{n+1}^{(j)} - u_{n-1}^{(j)}) + V_n u_n^{(j)}$$

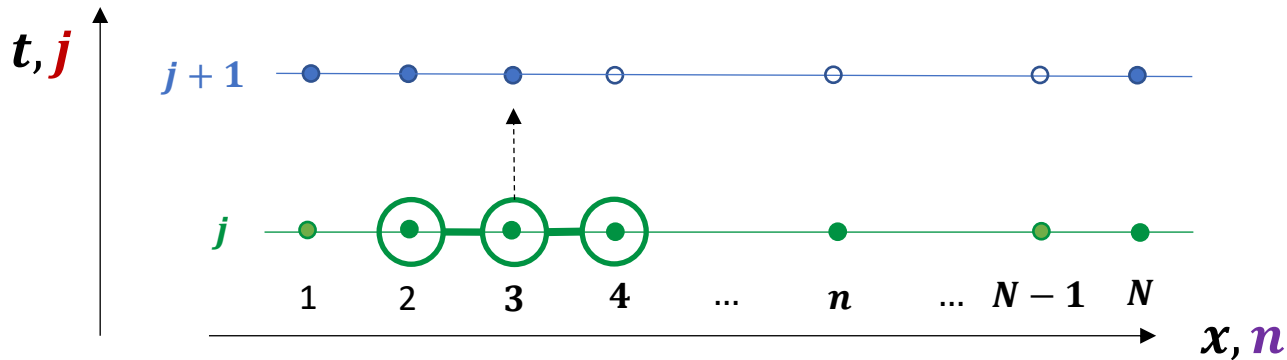
- Let $a = \frac{\Delta t}{(\Delta x)^2}$, $b_n = i\Delta t V_n$

$$u_n^{(j+1)} = (1 - i2a)u_n^{(j)} + ia(u_{n+1}^{(j)} + u_{n-1}^{(j)}) - b_n u_n^{(j)}$$

Forward-Time Centred-Space method (FTCS)

$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

$$u_n^{(j+1)} = (1 - i2a)u_n^{(j)} + ia \left(u_{n+1}^{(j)} + u_{n-1}^{(j)} \right) - b_n u_n^{(j)}$$



- Requires $O(N)$ calculations per iteration.
- => Computational time $\sim O(N) \times \text{number_of_iterations}$

Forward-Time Centred-Space method (FTCS)

$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

$$u_n^{(j+1)} = (1 - i2a)u_n^{(j)} + ia \left(u_{n+1}^{(j)} + u_{n-1}^{(j)} \right) - b_n u_n^{(j)}$$

- **Stability?**
- Consider the case without a potential (i.e. $b_n = 0$), split the solution into exact and error: $u_n^{(j)} = \mathbf{V}_n^{(j)} + \epsilon_n^{(j)}$. Substitute in the above equation and obtain for the error:

$$\epsilon_n^{(j+1)} = (1 - i2a)\epsilon_n^{(j)} + ia \left(\epsilon_{n+1}^{(j)} + \epsilon_{n-1}^{(j)} \right)$$

- Use Discrete Fourier Transform for space coordinate:

$$\epsilon_n^{(j)} = \sum_{k=0}^{N-1} E_k^{(j)} \exp(iq_k n) \quad q_k = \left(\frac{2\pi}{N} \right) k, k = 0, 1, \dots, N-1$$

Forward-Time Centred-Space method (FTCS)

$$\epsilon_n^{(j+1)} = (1 - i2a)\epsilon_n^{(j)} + ia(\epsilon_{n+1}^{(j)} + \epsilon_{n-1}^{(j)})$$

$$\epsilon_n^{(j)} = \sum_{k=0}^{N-1} E_k^{(j)} \exp(iq_k n) \quad q_k = \left(\frac{2\pi}{N}\right)k, k = 0, 1, \dots, N-1$$

- **Note:** $\epsilon_{n\pm 1}^{(j)} = \sum_{k=0}^{N-1} E_k^{(j)} \exp(iq_k(n \pm 1)) = \exp(iq_k(\pm 1)) \cdot \epsilon_n^{(j)}$
- Substitute:
$$E_k^{(j+1)} = (1 - i2a)E_k^{(j)} + ia(e^{iq_k} + e^{-iq_k})E_k^{(j)}$$
$$= [1 - i2a + i2a \cos(q_k)]E_k^{(j)} = \left[1 - i4a \sin^2\left(\frac{q_k}{2}\right)\right] E_k^{(j)}$$

$$E_k^{(j+1)} = \left[1 - i4a \sin^2\left(\frac{q_k}{2}\right)\right] E_k^{(j)}$$

- Stability: $g_k = \left|1 - i4a \sin\left(\frac{q_k}{2}\right)\right| = \sqrt{1 + 16a^2 \sin^4\left(\frac{q_k}{2}\right)} \geq 1$



This scheme is unstable for any value of the discretisation parameter

$$a = \frac{\Delta t}{(\Delta x)^2}$$

Forward-Time Centred-Space method (FTCS)

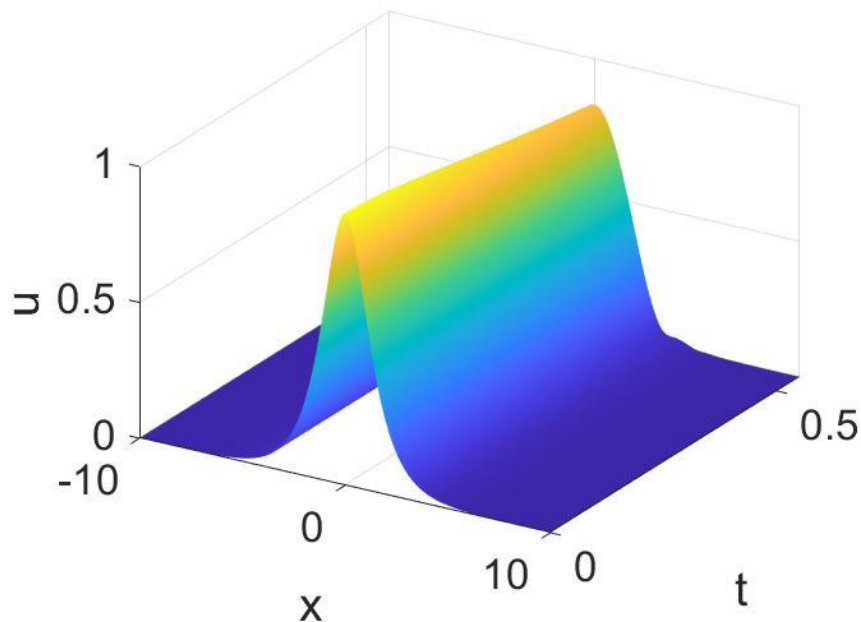
- Instability evolution:

Select $dx = 0.1$, $dt = 0.01$ ($a = 0.1$)

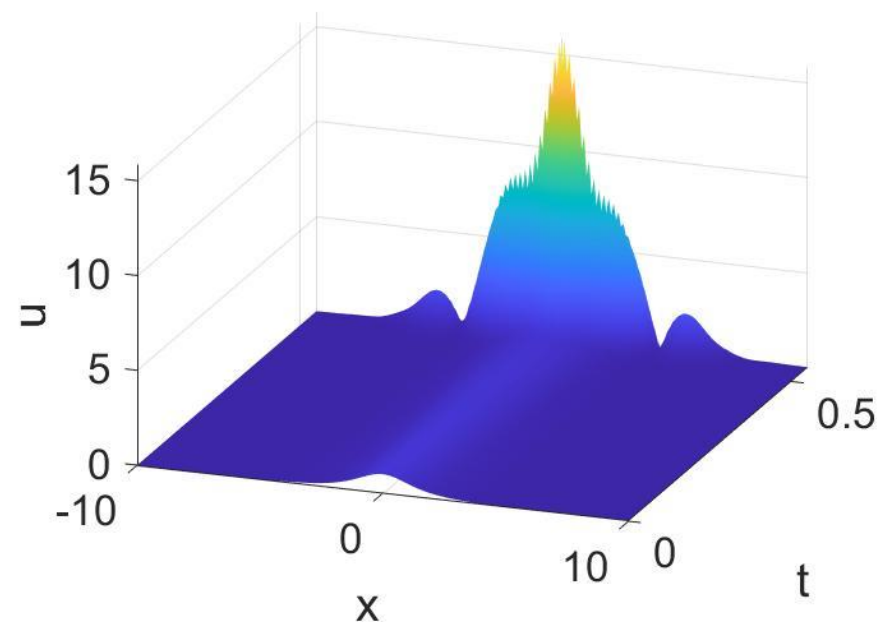
$V(x) = 0$

Initial condition: $u(x, t = 0) = \text{sech}(x)$

Exact solution



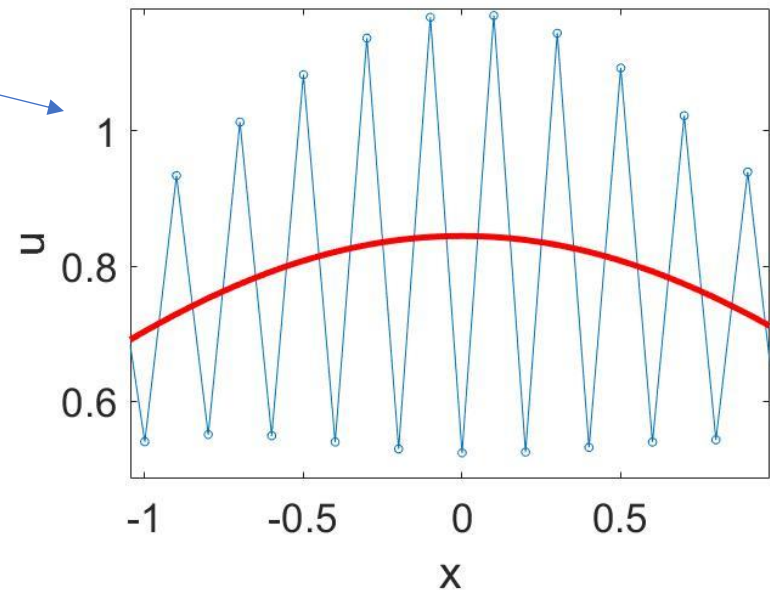
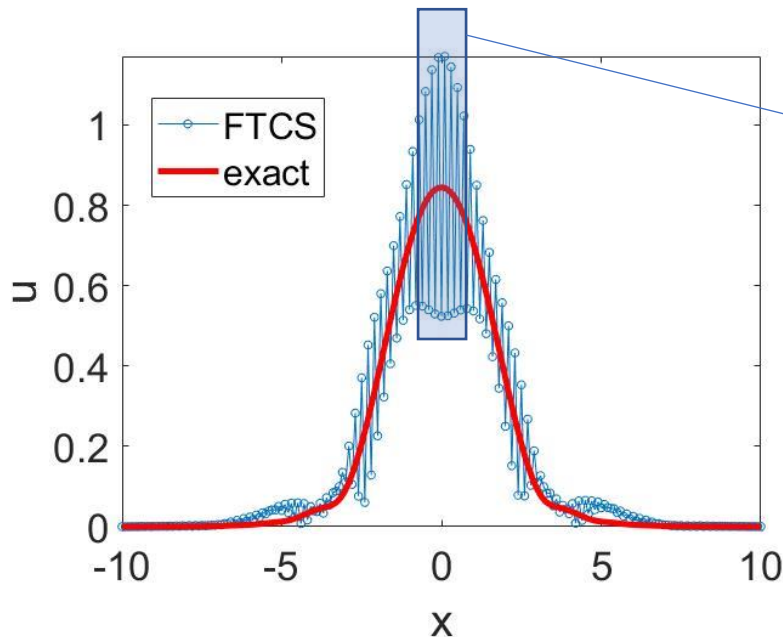
FTCS solution



Forward-Time Centred-Space method (FTCS)

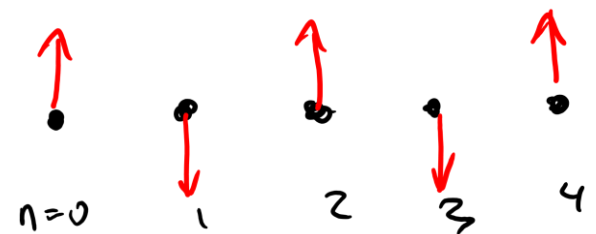
$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

- A snapshot at $t=0.5$



- Compare with stability analysis: $g_k = \sqrt{1 + 16a^2 \sin^2\left(\frac{q_k}{2}\right)}$

Largest growth for $q_k = \pi$, i.e. $\epsilon_n \sim \exp(i\pi n)$



Stability of FTCS methods

- More generally could replace simple Euler time iterations with a higher-order Runge-Kutta method
- Stability of any explicit method will generally impose a condition on the discretisation parameter:

$$a = \frac{\Delta t}{(\Delta x)^2} \leq C$$

This is known as Courant–Friedrichs–Lewy condition



Richard Courant
1888-1972

https://en.wikipedia.org/wiki/Richard_Courant



Kurt Otto Friedrichs
1901-1982

https://en.wikipedia.org/wiki/Kurt_Otto_Friedrichs



Hans Lewy
1904-1988

https://en.wikipedia.org/wiki/Hans_Lewy

Stability of FTCS methods

- More generally could replace simple Euler time iterations with a higher-order Runge-Kutta method
- Stability of any explicit method will generally impose a condition on the discretisation parameter:

$$a = \frac{\Delta t}{(\Delta x)^2} \leq C$$

This is known as Courant–Friedrichs–Lewy condition

- C is a specific constant, depends on a particular time-propagation explicit method, and a particular PDE. This is known as Courant number
- **Note:** the above condition locks together spatial (Δx) and temporal (Δt) resolutions. if you need to improve spatial resolution e.g. $\Delta x \rightarrow \Delta x/2$, you need to decrease the time step accordingly: $\Delta t \rightarrow \Delta t/4$

Crank-Nicolson Scheme

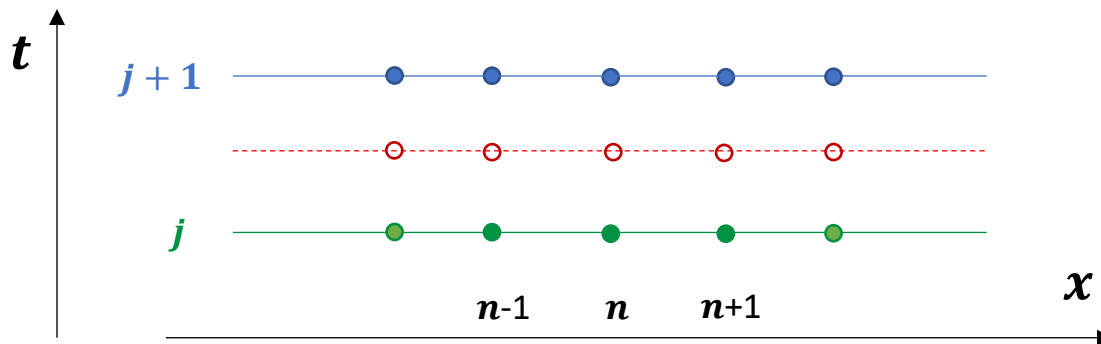
$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

- The FDA formula for the time derivative at $t = t_j$

$$i \frac{1}{\Delta t} [u_n^{(j+1)} - u_n^{(j)}] = \frac{1}{(\Delta x)^2} (2u_n^{(j)} - u_{n+1}^{(j)} - u_{n-1}^{(j)}) + V_n u_n^{(j)}$$

could be considered as a CDA formula for $\partial u / \partial t$ at
(non-existing in the grid) $t = t_{j+1/2}$

- But then we would need to evaluate the right-hand side at this non-existing time layer $j + 1/2$



- Can replace it with the average of layers j and $j + 1$, e.g.:

$$V_n u_n^{(j)} \rightarrow V_n u_n^{(j+1/2)} \approx \frac{1}{2} V_n (u_n^{(j+1)} + u_n^{(j)})$$



John Crank
1916-2006

https://en.wikipedia.org/wiki/John_Crank



Phyllis Nicolson
1917-1968

https://en.wikipedia.org/wiki/Phyllis_Nicolson

Crank-Nicolson Scheme

$$i \left[u_n^{(j+1)} - u_n^{(j)} \right] = a \left(2u_n^{(j)} - u_{n+1}^{(j)} - u_{n-1}^{(j)} \right) + \Delta t V_n u_n^{(j)} \quad a = \frac{\Delta t}{(\Delta x)^2}$$

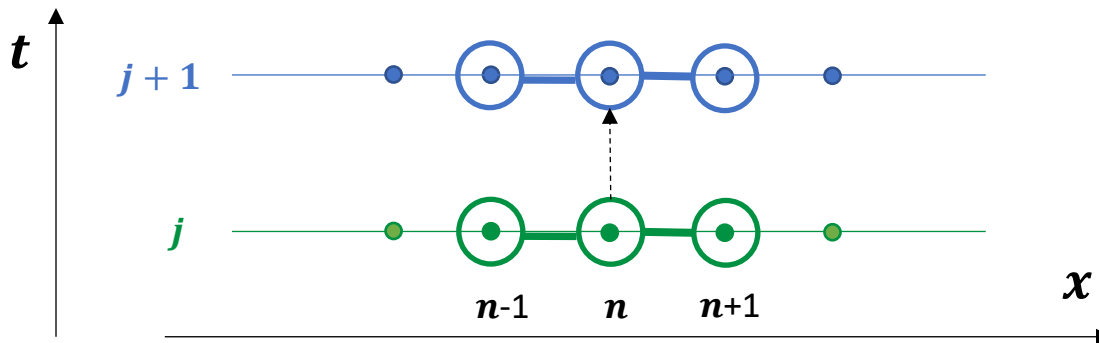


$$i \left[u_n^{(j+1)} - u_n^{(j)} \right] = \frac{1}{2} \left[a \left(2u_n^{(j)} - u_{n+1}^{(j)} - u_{n-1}^{(j)} \right) + \Delta t V_n u_n^{(j)} \right] + \frac{1}{2} \left[a \left(2u_n^{(j+1)} - u_{n+1}^{(j+1)} - u_{n-1}^{(j+1)} \right) + \Delta t V_n u_n^{(j+1)} \right]$$

- Note: we have (still unknown) values of $u_n^{(j+1)}$ in the right-hand side => **This is an implicit scheme**

- Can re-write as:

$$\left(i - a - \frac{\Delta t}{2} V_n \right) u_n^{(j+1)} + \frac{a}{2} \left(u_{n+1}^{(j+1)} + u_{n-1}^{(j+1)} \right) = \left(i + a + \frac{\Delta t}{2} V_n \right) u_n^{(j)} - \frac{a}{2} \left(u_{n+1}^{(j)} + u_{n-1}^{(j)} \right)$$



Crank-Nicolson Scheme

$$\left(i - a - \frac{\Delta t}{2} V_n\right) u_n^{(j+1)} + \frac{a}{2} \left(u_{n+1}^{(j+1)} + u_{n-1}^{(j+1)}\right) = \left(i + a + \frac{\Delta t}{2} V_n\right) u_n^{(j)} - \frac{a}{2} \left(u_{n+1}^{(j)} + u_{n-1}^{(j)}\right)$$

- In the vector format: $\hat{A} \vec{u}^{(j+1)} = \hat{B} \vec{u}^{(j)}$

$$\vec{u}^{(j+1)} = \hat{A}^{-1} \hat{B} \vec{u}^{(j)}$$

Note: \hat{A} is a tri-diagonal matrix. It does not require much effort to invert it.

- Stability?
- Repeating similar procedure as for FTCS, we can obtain for the error:

$$E_k^{(j+1)} = \left[\frac{1 - i2a \sin^2 \left(\frac{q_k}{2} \right)}{1 + i2a \sin^2 \left(\frac{q_k}{2} \right)} \right] E_k^{(j)}$$

$$|g_k| = \left| 1 - i2a \sin \left(\frac{q_k}{2} \right) \right| / \left| 1 + i2a \sin \left(\frac{q_k}{2} \right) \right| = 1 \quad \Rightarrow \quad \text{Marginally stable}$$

Crank-Nicolson Scheme

- Stable evolution:

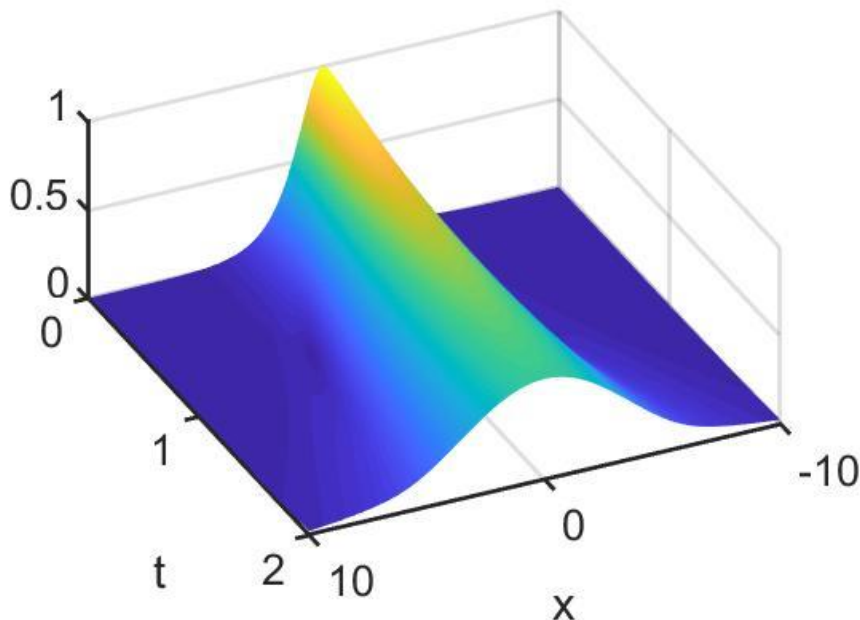
Select $dx = 0.1, dt = 0.02$ ($a=2$)

Initial condition: $u(x, t = 0) = \text{sech}(x)$

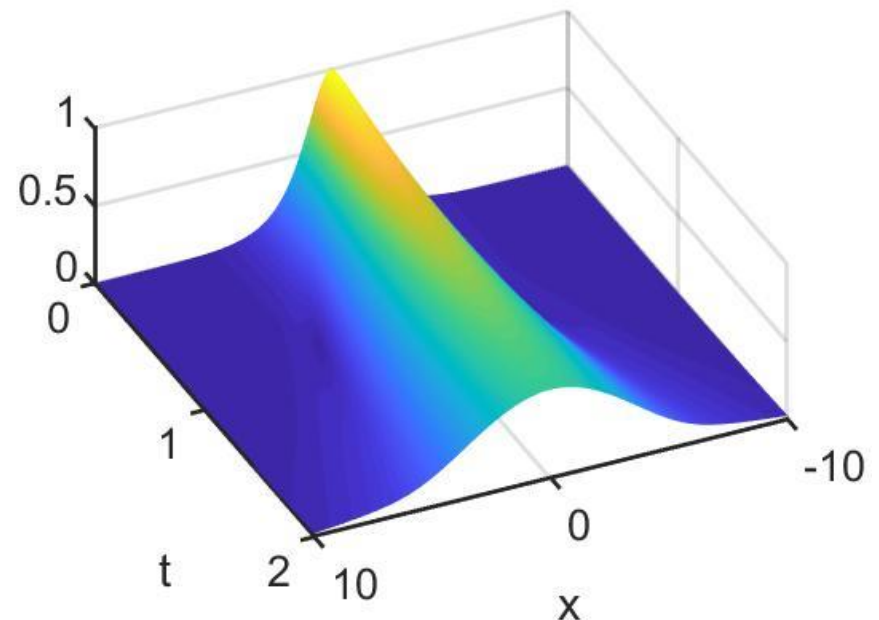
$$i \frac{\partial}{\partial t} u(x, t) = \left(-\frac{\partial^2}{\partial x^2} + V(x) \right) u(x, t)$$

$$V(x) = 0$$

Exact solution



CN solution



How this topic will be assessed:

- Coursework
- Typical questions on exam:

For a given (1+1)D PDE:

- *Discretise using FTCS scheme, write the resulting equation for a generic grid point away from the boundaries in a form suitable for iterations;*
- *Write down the equations for the boundary grid points: demonstrate how the BVP in spatial coordinate is implemented;*
- *Discuss (without performing a stability analysis) what is Courant-Freidrichs-Lewy criterion is, why is it important, what are the consequences of it for the spatial and time grid parameters.*

See further examples in worksheet 2