Numerical Simulations

ELEC2210: Applied Electromagnetics
ELEC2211: Electromechanical Energy Conversion
ELEC2219: Electromagnetism for EEE

Pt 2 Electrical Eng, Mechatronic Eng, EEE
Mathematical Modelling

- Physics, Chemistry, Biology, etc.
  - model of phenomenon results in set of differential or integral equations
Mathematical Modelling

- **Numerical Maths**
  - Discretization of space and time in order to make the problem solvable on computers
  - Choice of most appropriate approximation
  - Development of a numerical algorithm for solving the set of equations
Mathematical Modelling

- Phenomenon
- Mathematical Model
  i.e. set of equations
- Numerical Model
  i.e. set of discrete equations

- Algorithm
- Program

- Programming
  - Implementing the algorithm into software
Numerical differentiation is used when the derivatives of a function $u(x)$ cannot be obtained analytically, e.g. $u(x)$ is specified by a set of data, or during solution of differential equations.
Approximation of Derivatives

Taylor expansion for a function \( u(x) \):

\[
\begin{align*}
    u_{i-1} &= u_i - hu_i' + \frac{h^2}{2} u_i'' - \frac{h^3}{6} u_i''' + \frac{h^4}{24} u_i^{(4)} + O(h^5) \\
    u_{i+1} &= u_i + hu_i' + \frac{h^2}{2} u_i'' + \frac{h^3}{6} u_i''' + \frac{h^4}{24} u_i^{(4)} + O(h^5)
\end{align*}
\]

Linear combination of \( u_{i-1}, u_i, u_{i+1} \) :

\[
\begin{align*}
    u_i' &= \frac{u_{i+1} - u_i}{h} + O(h) = \frac{u_i - u_{i-1}}{h} + O(h) = \frac{u_{i+1} - u_{i-1}}{2h} + O(h^2) \\
    u_i'' &= \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + O(h^2)
\end{align*}
\]
Numerical integration – trapezium formula

\[ J = \int_{a}^{b} f(x) \, dx \]

Integral is the area under the curve:

\[ J \approx \sum_{i} \frac{1}{2} h_{i+1/2} (f_i + f_{i+1}) \]

with \( f_i \) taken at the edges of intervals.

\[ h_{i-1/2} = \frac{h_{i-1} + h_i}{2} \]
Numerical integration – Formula of Averages

\[ J \approx \sum_i \frac{1}{2} h_i f_i \]

with \( f_i \) taken at the centre of intervals.

Estimation of error:

\[
J = \sum_{i}^{x_{i+1/2}} \int_{x_{i-1/2}}^{x_i} f(x) \, dx = \sum_i f(x_i) h_i + \sum_i \frac{(h_i)^3}{8} \frac{1}{3} f''(x_i) + O\left(\max_i [h_i]^4\right)
\]

The total error is

\[ O\left((b-a) \max_i \left[f''(x_i) h_i^2\right]\right) \sim O(h^2) \]
Numerical integration – Accuracy Improvement

- Formulas for both methods are identical at the internal nodes
- Only edges are different
- Idea for an improvement!

Euler-Maclaurin formula:

\[ J = h \sum_i f(x_i) + \frac{h^2}{24} \left( f'(b) - f'(a) \right) + O(h^4) \]

with \( f_i \) taken at the centre of intervals.
Field Equations

Laplace’s equation: \( \nabla^2 V = 0 \)
\[ \nabla^2 A = 0 \]

Poisson’s equation: \( \nabla^2 V = f \)
\[ \nabla^2 A = F \]

Helmholtz equation: \( \nabla^2 V + k^2 V = 0 \) \hspace{1cm} \text{homogeneous}
\[ \nabla^2 V + k^2 V = f \] \hspace{1cm} \text{non-homogeneous}

Diffusion equation:
\[ \nabla^2 V = \frac{1}{h^2} \frac{\partial V}{\partial t} \]

Wave equation:
\[ \nabla^2 V = \beta^2 \frac{\partial^2 V}{\partial t^2} \]
Laplace’s Equation

Scalar field:
\[ \nabla^2 V = 0 \]

Vector field:
\[ \nabla^2 A = 0 \]
\[ \nabla^2 A_x = 0 \]
\[ \nabla^2 A_y = 0 \]
\[ \nabla^2 A_z = 0 \]

\[ \nabla^2 A_x = \frac{\partial^2 A_x}{\partial x^2} + \frac{\partial^2 A_x}{\partial y^2} + \frac{\partial^2 A_x}{\partial z^2} = 0 \]

Similar expressions for y and z components.
Boundary Conditions

- Infinite number of solutions for differential equation
- Set of additional restraints to make the solution unique
- Behaviour of the function at the boundary

- Value to the function is given at the boundary
  - Dirichlet boundary condition
- Value of the normal derivative of the function is given at the boundary
  - Neumann boundary condition
- Mixed problems

\[ \nabla^2 V = 0 \]

Differential Eq. inside

Boundary value at the edge

\[ V\big|_G = \varphi(x, y, z) \]

\[ \frac{\partial V}{\partial n}\big|_G = \psi(x, y, z) \]
Example

- **Steady state current flow problem**

  \[ \sigma = 1 \text{ S/m} \]

  \[ \mathbf{\nabla} \mathbf{V} = \sigma \cdot \mathbf{J} \]

  \[ \mathbf{J} = -\sigma \cdot \mathbf{\nabla} \mathbf{V} \]

  \[ J_n = \text{value or } V = \text{value} \]

  \[ J_n = 0 \quad V = 0 \]
Effect of boundary conditions

- Steady state current flow problem:
- Total current is the same for both cases

\[ V = \text{value} \quad \text{and} \quad J_n = \text{value} \]
Other Types

Poisson’s equation:
\[ \nabla^2 V = f \]
\[ \nabla^2 A = F \]

Helmholtz:
\[ \nabla^2 V + k^2 V = 0 \]
\[ \nabla^2 V + k^2 V = f \]
### Symmetry

- **Identical problems**

\[ J = -\sigma \cdot \text{grad}V \]

\( J_n = \text{value} \) or \( V = \text{value} \)

\( J_n = 0 \)
- Can be applied if
  - Geometry has some symmetry
  - Boundary conditions follow this symmetry
  - Material properties are uniform or also follow this symmetry

\[ J_n = \text{value} \quad \text{or} \quad V = \text{value} \]

\[ J_n = 0 \]

\[ V = 0 \]
The finite-difference method

$\Phi$ or $\frac{\partial \Phi}{\partial n}$ defined everywhere on the boundary

Equation specified inside:

$\nabla^2 \Phi = 0$

A unique solution exists!
The finite-difference method

Laplace's equation in 2D: \( \nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \)

\[
\Phi_{i+1,j} = \Phi_{i,j} + \Delta x \frac{\partial \Phi}{\partial x}_{i,j} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 \Phi}{\partial x^2}_{i,j} + \frac{(\Delta x)^3}{3!} \frac{\partial^3 \Phi}{\partial x^3}_{i,j} + \ldots
\]

\[
\Phi_{i-1,j} = \Phi_{i,j} - \Delta x \frac{\partial \Phi}{\partial x}_{i,j} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 \Phi}{\partial x^2}_{i,j} - \frac{(\Delta x)^3}{3!} \frac{\partial^3 \Phi}{\partial x^3}_{i,j} + \ldots
\]

Adding:

\[
\Phi_{i+1,j} + \Phi_{i-1,j} = 2\Phi_{i,j} + (\Delta x)^2 \frac{\partial^2 \Phi}{\partial x^2}_{i,j} + O\left\{ (\Delta x)^4 \right\}
\]

\[
\frac{\partial^2 \Phi}{\partial x^2}_{i,j} = \frac{\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}}{(\Delta x)^2}
\]
The finite-difference method

Laplace’s equation in 2D: \( \nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \)

\[
\left. \frac{\partial^2 \Phi}{\partial x^2} \right|_{i,j} = \frac{\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}}{(\Delta x)^2}
\]

Similarly, under the same assumptions, in the y direction:

\[
\left. \frac{\partial^2 \Phi}{\partial y^2} \right|_{i,j} = \frac{\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}}{(\Delta y)^2}
\]

\[
\frac{1}{(\Delta x)^2} \left( \Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j} \right) + \frac{1}{(\Delta y)^2} \left( \Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1} \right) = 0
\]
The finite-difference method

Laplace's equation in 2D:

\[ \nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \]

\[
\frac{1}{(\Delta x)^2} \left( \Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j} \right) + \frac{1}{(\Delta y)^2} \left( \Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1} \right) = 0
\]

If for convenience we choose a square mesh, so that \( \Delta x = \Delta y \):

\[
\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = 0
\]

or

\[
\Phi_{i,j} = \frac{1}{4} \left( \Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} \right)
\]
The finite-difference method

Laplace's equation in 2D:

\[ \nabla^2 \Phi = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = 0 \]

\[
\frac{1}{(\Delta x)^2} (\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}) + \frac{1}{(\Delta y)^2} (\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}) = 0
\]

If for convenience we choose a square mesh, so that \( \Delta x = \Delta y \):

\[
\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = 0
\]

or

\[
\Phi_{i,j} = \frac{1}{4} \left( \Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} \right)
\]

A five-point computation scheme:
Example

\[ V = 100 \text{ V} \]

\[ V = 0 \]
Example
Example

\[ V_{i,j} = \frac{1}{4}(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}) \]

\[
\begin{align*}
V_2 + 100 + V_2 + 0 - 4V_1 &= 0 \\
V_1 + 100 + V_3 + 0 - 4V_2 &= 0 \\
V_2 + 100 + V_4 + 0 - 4V_3 &= 0 \\
V_3 + 100 + 100 + V_5 - 4V_4 &= 0 \\
0 + V_4 + 100 + V_4 - 4V_5 &= 0 \\
\end{align*}
\]

\[
\begin{bmatrix}
4 & -2 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & -2 & 4
\end{bmatrix} \begin{bmatrix}
V_1 \\
V_2 \\
V_3 \\
V_4 \\
V_5
\end{bmatrix} = \begin{bmatrix}
100 \\
100 \\
100 \\
200 \\
100
\end{bmatrix}
\]

\[ Ax = b \]
Jacobi Method

\[ V_2 + 100 + V_2 + 0 - 4V_1 = 0 \]
\[ V_1 + 100 + V_3 + 0 - 4V_2 = 0 \]
\[ V_2 + 100 + V_4 + 0 - 4V_3 = 0 \]
\[ V_3 + 100 + 100 + V_5 - 4V_4 = 0 \]
\[ 0 + V_4 + 100 + V_4 - 4V_5 = 0 \]

\[ V_1 = \frac{1}{4} (2V_2 + 100) \]
\[ V_2 = \frac{1}{4} (V_1 + V_3 + 100) \]
\[ V_3 = \frac{1}{4} (V_2 + V_4 + 100) \]
\[ V_4 = \frac{1}{4} (V_3 + V_5 + 200) \]
\[ V_5 = \frac{1}{4} (2V_4 + 100) \]
Gauss-Seidel Method

\[
V_{i,j}^{(k+1)} = \frac{1}{4} \left( V_{i-1,j}^{(k+1)} + V_{i,j-1}^{(k+1)} + V_{i+1,j}^{k} + V_{i,j+1}^{k} \right)
\]
Successive Over-Relaxation (SOR)

\[
V_{i,j}^{(k+1)} = \frac{1}{4} \left( V_{i-1,j}^{(k+1)} + V_{i,j-1}^{(k+1)} + V_{i+1,j}^{(k+1)} + V_{i,j+1}^{(k+1)} \right)
\]

where \( k \) is an iteration ‘count’

Let

\[
\delta_{i,j}^{(k+1)} = V_{i,j}^{(k+1)} - V_{i,j}^{k}
\]

where \( \delta \) is a ‘residual’

or

\[
V_{i,j}^{(k+1)} = V_{i,j}^{k} + \delta_{i,j}^{(k+1)}
\]

Let

\[
V_{i,j}^{(k+1)} = V_{i,j}^{k} + \alpha \delta_{i,j}^{(k+1)}
\]

\( 1 \leq \alpha < 2 \)

or

\[
V_{i,j}^{(k+1)} = \left(1 - \alpha\right)V_{i,j}^{k} + \alpha \frac{1}{4} \left( V_{i-1,j}^{(k+1)} + V_{i,j-1}^{(k+1)} + V_{i+1,j}^{(k+1)} + V_{i,j+1}^{(k+1)} \right)
\]

or

\[
V_{i,j}^{(k+1)} = \left(1 - \alpha\right)V_{i,j}^{k} + \alpha \frac{1}{4} \left( V_{i-1,j}^{(k+1)} + V_{i,j-1}^{(k+1)} + V_{i+1,j}^{(k+1)} + V_{i,j+1}^{(k+1)} \right)
\]

over-relaxation

\[
\text{five-point formula}
\]

Note: \( \alpha=1 \) for Gauss-Seidel
When to stop iterations?

\[
V_{i,j}^{(k+1)} = \frac{1}{4}\left(V_{i-1,j}^{(k+1)} + V_{i,j-1}^{(k+1)} + V_{i+1,j}^k + V_{i,j+1}^k\right)
\]

\[
\delta_{i,j}^{(k+1)} = V_{i,j}^{(k+1)} - V_{i,j}^k
\]

\[
\varepsilon_{local} = \frac{\max_{i,j}\left|\delta_{i,j}^{(k+1)}\right|}{V_{\text{max}}} \times 100 \leq \text{local criterion} \quad (e.g. 1\%)
\]

\[
\varepsilon_{global} = \frac{1}{M \times N} \sum_{i,j}\left|\delta_{i,j}^{(k+1)}\right| \times 100 \leq \text{global criterion} \quad (e.g. 0.5\%)
\]
Remarks

- **Note:**
  - Other methods of solving $Ax=b$ exist
  - SOR particularly easy to implement in FD
  - Matrix sparsity may be explored
  - Matrix symmetry may be an issue

\[
\begin{bmatrix}
4 & -2 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 \\
0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & -2 & 4 \\
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2 \\
V_3 \\
V_4 \\
V_5 \\
\end{bmatrix}
= 
\begin{bmatrix}
100 \\
100 \\
100 \\
200 \\
100 \\
\end{bmatrix}
\]

\[V = 100 \text{ V}\]

\[V = 0\]
Advantages & Disadvantages of Finite Difference Method

- **Pros:**
  - intuitive and easy to implement
  - SOR natural for a ‘five point scheme’ (Laplace)
  - writing a code relatively straightforward

- **Cons:**
  - Often non-uniform mesh is needed
    - uniform grid difficult to fit into practical devices
    - curved boundaries awkward to handle
    - mesh grading difficult
  - Boundary conditions require special schemes (e.g. conservation laws to ensure charge conservation or Gauss’s Law)
The finite element method (FEM)

**Pros:**
- FE mesh easy to fit into practical devices
- Curved boundaries simple to handle
- Mesh grading possible and effective
- Boundary conditions naturally satisfied

**Cons:**
- Less intuitive
- Much more difficult to implement
- Writing a code challenging
Consider a single element and the following approximating polynomial

\[ V = a + bx + cy + dxy + ex^2 + fy^2 + ... \]

We choose as many terms as there are ‘nodes’ in the element:

**First order triangle** \( V = a + bx + cy \)

**Second order triangle** \( V = a + bx + cy + dxy + ex^2 + fy^2 \)

**Rectangle** \( V = a + bx + cy + dxy \)
First order triangular element

In the three vertices (nodes) the potential is

\[ V_1 = a + bx_1 + cy_1 \]
\[ V_2 = a + bx_2 + cy_2 \]
\[ V_3 = a + bx_3 + cy_3 \]

or

\[
\begin{bmatrix}
V_1 \\
V_2 \\
V_3
\end{bmatrix}
= 
\begin{bmatrix}
1 & x_1 & y_1 \\
1 & x_2 & y_2 \\
1 & x_3 & y_3
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
\]

and rearranging

\[
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
1 & x_1 & y_1 \\
1 & x_2 & y_2 \\
1 & x_3 & y_3
\end{bmatrix}^{-1}
\begin{bmatrix}
V_1 \\
V_2 \\
V_3
\end{bmatrix}
\]

and substituting back yields:

\[
V = \begin{bmatrix}
1 & x & y
\end{bmatrix}
\begin{bmatrix}
1 & x_1 & y_1 \\
1 & x_2 & y_2 \\
1 & x_3 & y_3
\end{bmatrix}^{-1}
\begin{bmatrix}
V_1 \\
V_2 \\
V_3
\end{bmatrix}
\]
or

\[
V = \sum_{i=1}^{3} V_i \alpha_i(x, y)
\]
Potential across an element

First order triangular element

\[
V = \sum_{i=1}^{3} V_i \alpha_i(x, y)
\]

where

\[
\alpha_1 = \frac{1}{2A} \left\{ (x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y \right\}
\]

\[
\alpha_2 = \frac{1}{2A} \left\{ (x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y \right\}
\]

\[
\alpha_3 = \frac{1}{2A} \left\{ (x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y \right\}
\]

At the vertices:

\[
\alpha_1(x_1, y_1) = \frac{1}{2A} \left\{ (x_2y_3 - x_3y_2) + (y_2 - y_3)x_1 + (x_3 - x_2)y_1 \right\} = \frac{2A}{2A} = 1
\]

\[
\alpha_1(x_2, y_2) = \frac{1}{2A} \left\{ (x_2y_3 - x_3y_2) + (y_2 - y_3)x_2 + (x_3 - x_2)y_2 \right\} = 0
\]

\[
\alpha_1(x_3, y_3) = \frac{1}{2A} \left\{ (x_2y_3 - x_3y_2) + (y_2 - y_3)x_3 + (x_3 - x_2)y_3 \right\} = 0
\]
Basis for 2D triangular element

First order triangular element

\[ V = \sum_{i=1}^{3} V_i \alpha_i(x, y) \]

where

\[ \alpha_1 = \frac{1}{2A} \{(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\} \]

\[ \alpha_2 = \frac{1}{2A} \{(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\} \]

\[ \alpha_3 = \frac{1}{2A} \{(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\} \]

In general

\[ \alpha_i(x_j, y_j) = 0 \quad i \neq j \]

\[ = 1 \quad i = j \]

Each function vanishes at all vertices but one, where it assumes the value of one.
Minimum Energy

Consider Laplacian equation in 2D for an electrostatic system

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

where the electric field \( \mathbf{E} \) is given by

$$\mathbf{E} = -\nabla V = -\nabla V = -\left\{ \frac{\partial V}{\partial x} \mathbf{i} + \frac{\partial V}{\partial y} \mathbf{j} \right\}$$

The stored field energy:

$$W = \frac{1}{2} \int_{\Omega} \mathbf{E} \cdot \nabla \mathbf{D} \, d\Omega = \frac{1}{2} \int_{\Omega} \mathbf{E} \cdot \mathbf{E} \, d\Omega = \frac{1}{2} \int_{\Omega} \varepsilon \left( \left[ \frac{\partial V}{\partial x} \right]^2 + \left[ \frac{\partial V}{\partial y} \right]^2 \right) \, d\Omega = \frac{1}{2} \int_{\Omega} \varepsilon |\nabla V|^2 \, d\Omega$$

where integration is carried out over a 2D region, and is thus taken per unit length/depth.

The principle of equilibrium requires that the potential distribution must be such as to minimise the stored field energy. This minimum-energy principle is mathematically equivalent to our original differential equation in the sense that the potential distribution which satisfies Laplace’s equation will also minimise the energy, and vice versa.
Energy stored in an element

**First order triangular element**

$$V = \sum_{i=1}^{3} V_i \alpha_i(x, y)$$

We can now associate energy with each element, and remembering that in 2D this energy will be taken per unit length/depth we write

$$W^{(e)} = \frac{1}{2} \int_{\epsilon} \varepsilon |\nabla V|^2 dS$$

where integration is performed over the element area.

Using \( V = \sum_{i=1}^{3} V_i \alpha_i(x, y) \) we find \( \nabla V = \sum_{i=1}^{3} V_i \nabla \alpha_i(x, y) \)

so that the element energy becomes

$$W^{(e)} = \frac{1}{2} \varepsilon \sum_{i=1}^{3} \sum_{j=1}^{3} V_i \left( \int_{\epsilon} \nabla \alpha_i \cdot \nabla \alpha_j dS \right) V_j$$

which may be written in a compact form

$$W^{(e)} = \frac{1}{2} \varepsilon [V]^T [N]^{(e)} [V]$$

where \([V]\) is the vector of vertex values of potential, the superscript \(T\) denotes transposition, and the \(3 \times 3\) square element matrix \([N]^{(e)}\) is defined by

$$N_{i,j}^{(e)} = \int_{\epsilon} \nabla \alpha_i \cdot \nabla \alpha_j dS$$
Energy stored in an element

**First order triangular element**

\[ W^{(e)} = \frac{1}{2} \varepsilon [V]^T [N]^{(e)} [V] \]

\[ N_{i,j}^{(e)} = \int_{e} \nabla \alpha_i \cdot \nabla \alpha_j \ dS \]

\[ \alpha_1 = \frac{1}{2A} \{(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y\} \]

\[ \nabla \alpha_1 = \frac{\partial \alpha_1}{\partial x} \mathbf{i} + \frac{\partial \alpha_1}{\partial y} \mathbf{j} = \frac{1}{2A} \{(y_2 - y_3)\mathbf{i} + (x_3 - x_2)\mathbf{j}\} \]

\[ \alpha_2 = \frac{1}{2A} \{(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y\} \]

\[ \nabla \alpha_2 = \frac{\partial \alpha_2}{\partial x} \mathbf{i} + \frac{\partial \alpha_2}{\partial y} \mathbf{j} = \frac{1}{2A} \{(y_3 - y_1)\mathbf{i} + (x_1 - x_3)\mathbf{j}\} \]

\[ \alpha_3 = \frac{1}{2A} \{(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y\} \]

\[ \nabla \alpha_3 = \frac{\partial \alpha_3}{\partial x} \mathbf{i} + \frac{\partial \alpha_3}{\partial y} \mathbf{j} = \frac{1}{2A} \{(y_1 - y_2)\mathbf{i} + (x_2 - x_1)\mathbf{j}\} \]
Energy stored in an element

First order triangular element

\[
W^{(e)} = \frac{1}{2} \varepsilon \left[ V \right]^T \left[ N^{(e)} \right] \left[ V \right]
\]

\[
N^{(e)}_{i,j} = \int_{e} \nabla \alpha_i \cdot \nabla \alpha_j \, dS
\]

\[
\nabla \alpha_1 = \frac{\partial \alpha_1}{\partial x} \mathbf{i} + \frac{\partial \alpha_1}{\partial y} \mathbf{j} = \frac{1}{2A} \left\{ (y_2 - y_3)\mathbf{i} + (x_3 - x_2)\mathbf{j} \right\}
\]

\[
\nabla \alpha_2 = \frac{\partial \alpha_2}{\partial x} \mathbf{i} + \frac{\partial \alpha_2}{\partial y} \mathbf{j} = \frac{1}{2A} \left\{ (y_3 - y_1)\mathbf{i} + (x_1 - x_3)\mathbf{j} \right\}
\]

\[
\nabla \alpha_3 = \frac{\partial \alpha_3}{\partial x} \mathbf{i} + \frac{\partial \alpha_3}{\partial y} \mathbf{j} = \frac{1}{2A} \left\{ (y_1 - y_2)\mathbf{i} + (x_2 - x_1)\mathbf{j} \right\}
\]

The scalar product of two vectors, say \( \mathbf{a} \) and \( \mathbf{b} \), in a Cartesian coordinate system in 2D:

\[
\mathbf{a} \cdot \mathbf{b} = (a_x \mathbf{i} + a_y \mathbf{j}) \cdot (b_x \mathbf{i} + b_y \mathbf{j})
\]

\[
= a_x b_x \mathbf{i} \cdot \mathbf{i} + a_x b_y \mathbf{i} \cdot \mathbf{j} + a_y b_x \mathbf{j} \cdot \mathbf{i} + a_y b_y \mathbf{j} \cdot \mathbf{j}
\]

\[
= a_x b_x + a_y b_y
\]

\[
V = \sum_{i=1}^{3} V_i \alpha_i (x, y)
\]

\[
N^{(e)}_{1,1} = \frac{1}{4A} \left\{ (y_2 - y_3)^2 + (x_3 - x_2)^2 \right\}
\]

\[
N^{(e)}_{1,2} = \frac{1}{4A} \left\{ (y_2 - y_3)(y_3 - y_1) + (x_3 - x_2)(x_1 - x_3) \right\}
\]

\[
N^{(e)}_{1,3} = \frac{1}{4A} \left\{ (y_2 - y_3)(y_1 - y_2) + (x_3 - x_2)(x_2 - x_1) \right\}
\]

\[
N^{(e)}_{2,1} = \frac{1}{4A} \left\{ (y_3 - y_1)(y_2 - y_3) + (x_1 - x_3)(x_3 - x_2) \right\}
\]

\[
N^{(e)}_{2,2} = \frac{1}{4A} \left\{ (y_3 - y_1)^2 + (x_1 - x_3)^2 \right\}
\]

\[
N^{(e)}_{2,3} = \frac{1}{4A} \left\{ (y_3 - y_1)(y_1 - y_2) + (x_1 - x_3)(x_2 - x_1) \right\}
\]

\[
N^{(e)}_{3,1} = \frac{1}{4A} \left\{ (y_1 - y_2)(y_2 - y_3) + (x_2 - x_1)(x_3 - x_2) \right\}
\]

\[
N^{(e)}_{3,2} = \frac{1}{4A} \left\{ (y_1 - y_2)(y_3 - y_1) + (x_2 - x_1)(x_1 - x_3) \right\}
\]

\[
N^{(e)}_{3,3} = \frac{1}{4A} \left\{ (y_1 - y_2)^2 + (x_2 - x_1)^2 \right\}
\]
Total Energy

First order triangular element

\[ W^{(e)} = \frac{1}{2} \varepsilon [V]^T [N]^{(e)} [V] \]

\[ V = \sum_{i=1}^{3} V_i \alpha_i(x, y) \]

This completes the specification for an arbitrary element in the finite-element mesh. The total energy associated with the entire region will be found as the sum of individual element energies:

\[ W = \sum_{\text{All elements}} W^{(e)} \]

When assembling elements we notice that some nodes will be shared between elements. The global matrix must reflect the way in which individual elements are linked.
Example

Let's use only 6 nodes!
Matrix assembly

- Different finite elements may contribute to the same matrix entry – add together!
- Not straightforward operation
- Automation is possible – computer algorithms

\[
[N] = \begin{bmatrix}
1 + \frac{1}{2} + \frac{1}{2} & -\frac{1}{2} - \frac{1}{2} & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} - \frac{1}{2} & 1 + \frac{1}{2} + 1 & -\frac{1}{2} - \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\
0 & -\frac{1}{2} + \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} - \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 \\
-\frac{1}{2} & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Minimum Energy

To minimise the total energy we must differentiate with respect to a typical value of $V_k$ and equate to zero:

$$\frac{\partial W}{\partial V_k} = 0$$

where $k$ refers to node numbers in the global numbering system. However, some nodes will have a ‘prescribed’ potential (e.g. on the boundary) and we can differentiate only with respect to ‘free’ nodes.

$$\frac{\partial W}{\partial [V_f]_k} = \frac{\partial}{\partial [V_f]_k} \left[ [V_f]^T [V_p]^T \begin{bmatrix} N_{ff} & N_{fp} \\ N_{pf} & N_{pp} \end{bmatrix} [V_f] \right] = 0$$

Differentiation with respect to the free potentials results in the following matrix equation:

$$\begin{bmatrix} N_{ff} & N_{fp} \\ N_{pf} & N_{pp} \end{bmatrix} [V_f] = 0$$

$$W = \frac{1}{2} \varepsilon [V]^T [N][V]$$
Equations to solve

\[
\begin{bmatrix}
[N_{ff}] & [N_{fp}]
\end{bmatrix}
\begin{bmatrix}
[V_f] \\
[V_p]
\end{bmatrix} = 0
\]

\[
[N_{ff}][V_f] = -[N_{fp}][V_p]
\]

Formal solution:

\[
[V_f] = -[N_{ff}]^{-1}[N_{fp}][V_p]
\]

In our example:

\[
\begin{bmatrix}
2 & -1 \\
-1 & 2
\end{bmatrix}
\begin{bmatrix}
V_1 \\
V_2
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & -0.5 & -0.5 \\
-0.5 & -0.5 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
V_3 \\
V_4 \\
V_5 \\
V_6
\end{bmatrix}
\]

\[
\begin{align*}
2V_1 - V_2 &= 50 \\
-V_1 + 2V_2 &= 50\end{align*}
\]

\[
\therefore V_1 = V_2 = 50
\]
System of Linear Equations (again!)

Final equation: \[
[N_{ff}] [V_f] = -[N_{fp}] [V_p]
\]

This is in the form: \[A \mathbf{x} = \mathbf{b}\]

Comments:
- \([N_{ff}]\) (or \(A\)) may be a very large matrix
- \(\mathbf{b}\) is given by sources (boundary conditions)
- \(A\) is normally ‘sparse’ (we cannot see this here)
- \(A\) is often symmetrical

\[
[N] = \begin{bmatrix}
2 & -1 & 0 & 0 & -0.5 & -0.5 \\
-1 & 2 & -0.5 & -0.5 & 0 & 0 \\
0 & -0.5 & 1 & 0 & -0.5 & 0 \\
0 & -0.5 & 0 & 1 & 0 & -0.5 \\
-0.5 & 0 & -0.5 & 0 & 1 & 0 \\
-0.5 & 0 & 0 & -0.5 & 0 & 1 \\
\end{bmatrix}
\]
The global matrix may be partitioned as follows:

\[
[N] = \begin{bmatrix}
[N_{pp}] & [N_{pf}] & [N_{pp}] \\
[N_{fp}] & [N_{ff}] & [N_{fp}] \\
[N_{pp}] & [N_{pf}] & [N_{pp}]
\end{bmatrix}
\]

The \([N_{ff}]\) sub-matrix is of particular interest:

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Similarities and differences between FD and FE methods

- both rely on discrete representation of the solution
- a grid or mesh of nodes is needed
- for better accuracy a fine grid/mesh is required
- both lead to a large system of equations $Ax=b$
- solution of this equation is computationally expensive
- local and global errors need to be monitored
- visualisation and post-processing challenging because of large amounts of data

But:

- FD solution defined on a set of nodes only, FE solution described everywhere
- Simple FD grid restricted because of parallel lines, FE mesh flexible

Note:

- if FD grid and FE mesh matching → identical final system of equations
Solving \( \mathbf{A} \mathbf{x} = \mathbf{b} \)

- All numerical schemes lead to \( \mathbf{A} \mathbf{x} = \mathbf{b} \)
- FD and FE formulations result in very large systems of equations
- The \( \mathbf{A} \) matrix is very sparse
- \( \mathbf{A} \) is often symmetrical
- Efficient storage systems are required
- Fast and accurate computation is needed

### Comparison of methods

**Test problem:**
- Poisson’s equation over a unit square
- uniform mesh of first order triangles with \( n \) points

<table>
<thead>
<tr>
<th>( n ) nodes</th>
<th>( m ) bandwidth</th>
<th>Gauss elimination</th>
<th>Gauss-Seidel ( m \times n \times \text{iter} )</th>
<th>SOR ( n \log n )</th>
<th>ICCG</th>
<th>operation count (time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>8</td>
<td>( m^2 n )</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>441</td>
<td>23</td>
<td>33</td>
<td>619</td>
<td>123</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>961</td>
<td>33</td>
<td>152</td>
<td>-</td>
<td>536</td>
<td>70</td>
<td>70</td>
</tr>
</tbody>
</table>
Comparison

1. Gauss – Seidel iterations
   \((\alpha = 1)\)

2. SOR with \(\alpha_{opt} = 1.885\)
Faster Convergence

$\alpha = 1$

$\alpha_{opt} = 1.885$

Iteration count: 10
Faster Convergence

\[ \alpha = 1 \quad \text{vs} \quad \alpha_{opt} = 1.885 \]

Prescribed accuracy achieved in 106 iterations.

Iteration count: 106
Faster Convergence

\[ \alpha = 1 \]

\[ \alpha_{opt} = 1.885 \]

Prescribed accuracy achieved in 690 iterations.

Prescribed accuracy achieved in 106 iterations.

Iteration count: 690
Effect of Initial Conditions

Assume a linear initial solution:  Final solution:

Prescribed accuracy achieved in 51 iterations (instead of 106 when a ‘zero’ solution was initially used).
Effect of Accuracy

Assume a linear initial solution:

Final solution:

5 iterations needed to keep the local error below 1%
24 iterations needed to keep the local error below 0.2%
51 iterations needed to keep the local error below 0.05%
81 iterations needed to keep the local error below 0.01%
A typical CAD system for electromagnetics