Introduction to CEM

Consider a general case of a scatterer immersed in an inhomogeneous, anisotropic medium, shown in Fig. 1 with magnetic and electric sources. Assuming a time-harmonic dependence e^{j\omega t}

we write Maxwell's equations as

\[ \begin{align*}
\nabla \times \vec{H} &= \vec{J} + j \omega \vec{E} \\
\nabla \times \vec{E} &= -\vec{H} - j \omega \vec{B} \\
\n\nabla \cdot (\vec{E} \times \vec{B}) &= \rho_f \\
\n\nabla \cdot (\vec{E} \times \vec{H}) &= 0
\end{align*} \]

From Maxwell-Ampère Law we have:

\[ \nabla \times (\vec{E} \times \vec{B}) = \vec{E} \times \vec{J} + j \omega \vec{E} \]

From Faraday Law we have:

\[ \nabla \times (\vec{E} \times \vec{E}) = -\vec{B} \times \vec{H} - j \omega \vec{B} \]

These are coupled differential equations, need to eliminate one of \( \vec{E}, \vec{H} \)

\[ \nabla \times (\vec{E} \times \vec{E}) = -\nabla \times (\vec{B} \times \vec{H}) - j \omega (\vec{B} \times \vec{E}) \]

\[ \Rightarrow \nabla \times (\vec{E} \times \vec{E}) + j \omega \vec{E} \times \vec{E} = -\nabla \times (\vec{B} \times \vec{H}) - j \omega \vec{B} \]

from MA law

\[ \Rightarrow \nabla \times (\vec{B} \times \vec{E}) = -\nabla \times (\vec{B} \times \vec{H}) - j \omega \vec{B} \]

or

\[ \nabla \times \vec{B} = -\nabla \times (\vec{B} \times \vec{H}) - j \omega \vec{B} \]

This is an inhomogeneous vector wave equation. It does not have a closed form solution; thus, it has to be solved numerically.

Note: that this equation can be written in general operator form:

\[ \mathbf{L} \mathbf{E} = \mathbf{g} \]

where

\[ \mathbf{L} = \nabla \times (\vec{B} \times \vec{E}) - j \omega \mathbf{E} \quad \text{differential operator} \]

\[ \mathbf{g} = -\nabla \times (\vec{B} \times \vec{H}) - j \omega \vec{J} \quad \text{source function} \]

Sources are on the right \((\vec{B}, \vec{J})\); consequences on the left \((\mathbf{E}, \mathbf{H})\)
Assume that the medium is homogenous isotropic $\Rightarrow$ the constitutive parameters are scalars: $\varepsilon \rightarrow \varepsilon$ $\sigma \rightarrow \sigma$

Closed-form solution is possible in this medium!!!

(a) How?

Step 0: Maxwell's equations
\[ \nabla \times \mathbf{D} = \mathbf{J} + \varepsilon \frac{\partial \mathbf{E}}{\partial t} \]
\[ \nabla \cdot \mathbf{B} = 0 \]

Step 1: Take care of Faraday's law and apply vector identity
\[ \nabla \times (\nabla \times \mathbf{E}) = \nabla (\nabla \cdot \mathbf{E}) - \nabla \times \mathbf{H} \cdot \nabla \times \mathbf{E} \]
\[ -\nabla^2 \mathbf{E} + \nabla (\nabla \cdot \mathbf{E}) = -\nabla \times \mathbf{H} - \nabla \times \mathbf{E} \times \mathbf{H} \]

Step 2: Replace $\nabla \times \mathbf{B}$ and $\nabla \cdot \mathbf{B}$ with Maxwell-Ampere & Gauss' laws
\[ \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \mathbf{M} \frac{\partial \mathbf{E}}{\partial t} \]
\[ \nabla \cdot \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \mathbf{M} \frac{\partial \mathbf{E}}{\partial t} \]

Step 3: Group sources on right fields on the left
\[ \nabla^2 \mathbf{E} + \omega^2 \varepsilon \mu \mathbf{E} = \mathbf{J} + \omega \mu \mathbf{J} \frac{\partial \mathbf{E}}{\partial t} + \frac{1}{2} \frac{\partial \mathbf{H}^2}{\partial t} \]
\[ \nabla^2 \mathbf{E} + \omega^2 \varepsilon \mu \mathbf{E} = \mathbf{J} + \omega \mu \mathbf{J} \frac{\partial \mathbf{E}}{\partial t} + \frac{1}{2} \frac{\partial \mathbf{H}^2}{\partial t} \]
\[ \mathbf{b} = \omega \mu_0 \mathbf{M} \mathbf{E} \]

Step 4: Solve for $\mathbf{E}$ fields: How?

Let's rewrite the vector wave equation for the homogenous isotropic linear medium
\[ \nabla^2 \mathbf{E} + \omega^2 \varepsilon \mu \mathbf{E} = \mathbf{J} + \omega \mu \mathbf{J} \frac{\partial \mathbf{E}}{\partial t} \text{ or } \mathbf{L} \mathbf{E} = \mathbf{g} \text{ in operator form} \]

where
\[ \mathbf{L} = \nabla^2 + \frac{1}{\varepsilon} \frac{\partial^2}{\partial t^2} \text{ - Linear 2nd order diff. operator} \]
\[ \mathbf{g} = \omega \mu \mathbf{J} \frac{\partial \mathbf{E}}{\partial t} + \omega \mu \mathbf{M} \frac{\partial \mathbf{E}}{\partial t} \text{ - Excitation/source function} \]

So how do we solve this equation?

From:

* System/Medium is Linear $\Rightarrow$ the same system for the point
* Governing Equations Linear $\Rightarrow$ source excitation (Dirac delta function) $\delta(x) \rightarrow \mathbf{h}(x)$

1. Find response of the same system for the point
2. The response for arbitrary excitation $\mathbf{g}$ will be the convolution integral of $\mathbf{g}$ and $\mathbf{h}$ i.e. $\mathbf{h}(x) * \mathbf{g}(x)$ over the domain of Hex sources

Function $\mathbf{\epsilon}(r) = \mathbf{G}(r, r')$ is Green's function
Step 5: Express the fields into a medium as:

\[
E = - \iiint \left( j_{\omega} \mathbf{J}(r) + \nabla \times \mathbf{H}(r) + \frac{1}{2} \nabla' \times \mathbf{E}'(r) \right) G_0(r, r') dr'
\]

where \( G_0(r, r') = \frac{e^{-j\kappa r}}{4\pi |r - r'|} \) is the scalar free space Greens function.

Notes:

1. After finding \( E' \), \( H' \) is obtained from Faraday's Law.

2. Free-space Greens function:
   
   1. \( H(x) = \frac{j}{2\kappa} e^{-j\kappa x} \)
   2. \( H(r) = \frac{j}{4} \nabla \times (k \mathbf{E}) \)
   3. \( H(r) = G_0(r, r') = \frac{e^{-j\kappa r}}{4\pi r} \); \( r = |r - r'| \)

Q) How can we classify EM Problems?
Classification of EM Problems

Governing Equation (Operator)

- Differential
- Integral
- Integro-Diff.

FEM
FE/BIE
NOM

Boundary Conditions

Math
- Operator
Physics
- Physical
- Approximate
- Natural
- Radiation

Dirichlet
Neumann
Mixed

\[ \phi(x, y) = 0 \]
\[ \frac{\partial \phi(r)}{\partial n} = 0 \]
\[ \phi(r) + h(r) \frac{\partial \phi}{\partial n} = 0 \]

- All shown in homogeneous form:
  \[ \phi(r) = p(r) \] in inhomogeneous
  Dirichlet B.C.

Notes

1. \[ L\phi = q \] → Describes the deterministic problem

2. \[ L\phi = \lambda \phi \] → Describes the non-deterministic or eigen-value problem
   \[ \lambda \rightarrow \text{eigenvalues} \]
   \[ \phi \rightarrow \text{eigen-functions} \]
Numerical Integration

- Exact integration is seldom possible in CEM, so we have to use an approximak way → numerical integration.
- Numerical integration is also known as numerical quadrature and integration formulak that yield the approximate solution is > quadrature.

Example: FE-EM (cavity backscattering)

For the FEM part: \( A^e E^e = g^e \)

\[ \begin{align*}
\mathbf{I}_i^e &= \int \mathbf{B}_i \mathbf{D}_i \, dV^e - \int \mathbf{B}_i \mathbf{B}_i^T \mathbf{D}_i \, dV^e \\
\mathbf{g}_i^e &= \int \mathbf{B}_i \cdot \mathbf{F}_i \, dV^e
\end{align*} \]

(2) What is NI?

Consider a real function \( f(x) \) defined over an integral \( x \in [a, b] \). Then

\[ I = \int_a^b f(x) \, dx \]

is a number that represents the surface area between the curve and abscissa.

Quadrature rules try to find a rectangle with the same surface area \( Q \) as \( I \); i.e., \( Q \approx I \)

Thus an \( M \)-point quadrature rule \( Q \) for the definite integral \( I = \int_a^b f(x) \, dx \) is an approximation of the form

\[ Q = (b - a) \sum_{k=1}^{M} \omega_k f(x_k) \]

\( \omega_k \rightarrow \) weights

\( x_k \rightarrow \) abscissae

\( \int \rightarrow \) defines the rule and are chosen such that \( Q \approx I \).
Euler's and mid-point rule

Given a real function \( f(x) \) defined over \( x \in [a, b] \) compute \( I = \int_a^b f(x) \, dx \n\).* Algorithm for coding implementation

[STEP1] divide (discrete, mesh) interval \( x \in [a, b] \) in \( N \) equal segments/sub-intervals (N+1 points including a, b)

\[
\Delta x = \frac{b - a}{N} \quad x_i = x_0 + i \cdot \Delta x \quad i = 0, 1, \ldots, N
\]

[STEP2] compute \( f(x_i) \) for \( i = 0, 1, \ldots, N \)

STEP3 Compute surface area \( A_i \):

for each rectangle on discrete grid

\[
A_i = \int_{x_{i-1}}^{x_i} f(x) \, dx 
\]

[STEP4] Sum all \( A_i \)'s to obtain:

\[
Q = \sum_{i=1}^{N} A_i = h \sum_{i=1}^{N} f(x_i) \approx I
\]

Discussion Points:

(dp1) This method is easy to implement yet results are very inaccurate

(dp2) Accuracy improves if we increase number of sampling points. Time ↑

(dp3) This rule belongs to the class of closed rules as it includes the bounds a and b.

(dp4) Mid-point integration rule is obtained if we compute the value of a point \( \bar{x} \) in the middle of each segment \( x_{i-1}, x_i \) :

\[
x_i = x_i - x_{i-1} \quad \text{and approximate our integral with}
\]

\[
Q = h \sum_{i=1}^{N} f \left( \frac{x_i + x_{i-1}}{2} \right) = h \sum_{i=1}^{N} f(x_i)
\]

(dp5) Mid-point integration rule is an open rule as values of function \( f(x) \) at \( x = a, b \) are not taken into account.

(dp6) Mid-point is yanked - dangerous for highly oscillatory functions.
- **Trapezoidal Rule:**

Given a real function \( f(x) \) defined over an interval \( x \in [a, b] \) find \( \int_a^b f(x) \, dx \)

**Algorithm**

**Step 1:** Discretize interval \( x \in [a, b] \) in \( N \) equal segments such that

\[
h = \Delta x = \frac{b-a}{N} \quad \text{and} \quad x_k = x_0 + kh, \quad k = 1, 2, \ldots, N
\]

**Step 2:** For \( k = 0, 1, \ldots, N \) compute \( f(x_k) \)

**Step 3:** Form trapezoidal areas \( A_k = \frac{h}{2} \left( f(x_{k-1}) + f(x_k) \right) \) and for each trapezoid, compute its surface area as

\[
A_k = \frac{h}{2} \left( f(x_{k-1}) + f(x_k) \right), \quad k = 1, 2, \ldots, N
\]

**Step 4:** Sum all \( A_k \)'s to obtain the trapezoidal rule:

\[
Q = \frac{h}{2} \sum_{k=1}^{N} \left( f(x_k) + f(x_{k-1}) \right) = \frac{h}{2} \left( f(a) + f(b) \right) + h \sum_{k=1}^{N-1} f(x_k)
\]

**Discussion Points**

1. Easy to implement
2. More accurate than Euler rule
3. This is a closed rule
**Simpson's Rule**

Given a real function \( f(x) \), defined over an interval \([a, b]\), compute \( Q = \int_a^b f(x) \, dx \).

**Algorithm**

**STEP 1** Discretize the interval \([a, b]\) into even number \( N = 2n \) segments.

\[ h = \frac{b - a}{N} \]

\[ x_i = a + i \cdot h \quad i = 0, \ldots, N \]

**STEP 2** For \( i = 0, \ldots, N \), compute \( f(x_i) \).

**STEP 3** Form a surface area:

\[ A_x = \frac{h}{3} \left( f(x_{i-2}) + 4f(x_{i-1}) + f(x_i) \right) \]

and compute the surface area:

\[ A_x = \sum_{i=1}^{N} A_x \]

**STEP 4** Sum all \( A_x \)’s:

\[ Q = \frac{h}{3} \left[ f(a) + 4f(b - h) + 2f(2h - 2h) + 4f(3h - 3h) + \ldots + f(b) \right] \]

**Discussion Points**

**dp1** There is error on pp. 197 (eq. 3.135) in Sadiku.

**dp2** Simpson's rule is more accurate than the previous 2 methods.

**dp3** This is a closed rule.

**dp4** Theorem: If \( f(x) \) and its first 4 derivatives are continuous on \([a, b]\), then the absolute error of numerical integration:

\[ |\int_a^b f(x) \, dx - Q_{5a}| \leq \frac{(b-a)^5}{2880} M_4 \]

where \( M_4 \) is an upper bound on \( |f^{(4)}(x)| \) on \([a, b]\).

- **Newton-Cotes Rules** (see Sadiku)

- Previous rules are special cases of NC rules.

- Order of polynomial approximation defines the number of subintervals so that the total number of segments is integer number of subintervals.
**Gauss Integration**

- Very efficient and accurate method.
- Non-uniform sampling points - Gauss.

**General Form of Gauss' Quadrature Rules** is:

\[
\int_{a}^{b} f(x) \, dx \approx \sum_{i=1}^{n} w_i f(x_i)
\]

- (a,b) interval at which a sequence of orthogonal polynomials \( \{w_i(x)\} \) exist.
- \( x_i \) - zeros of \( w_i(x) \).
- \( w_i(x) \) - orthogonal polynomials such that Gauss approximation is of \( 2n-1 \) degree of precision.

**Note:** A system of functions \( \Phi_n \) \( (n=0,1,...) \) is said to be orthogonal with weight \( \omega(x) \) on the interval \( (a,b) \) if

\[
\int_{a}^{b} \omega(x) \Phi_n(x) \Phi_m(x) \, dx = 0 \quad \text{for} \quad n \neq m
\]

- We will work only with Gauss Legendre quadratures and refer to it as a Gaussian Integration.

- So for a given function \( f(x) \) defined over \( [a,b] \) we have:

\[
\int_{a}^{b} f(x) \, dx = \frac{b-a}{2} \int_{-1}^{1} f(\xi) \, d\xi \approx \frac{b-a}{2} \sum_{i=1}^{n} w_i f(\xi_i)
\]

- \( n \) - number of sampling points that generate \( n \)-weights \( w_i : i=1, \ldots, n \).
Objective: Find weights and abscissae!  

Algorithm: Set a group (set) of "defining equations" to obtain 2n x 2n system of 2n equations with 2n unknowns.

Ex. One point (n=1) Gauss Integration

\[ \int_{-1}^{1} f(\xi) \, d\xi = w_1 f(\xi_1) \]

Need 2 equations!!

1st defining equation: 0th order polynomial \( f(\xi) = 1 \)

\[ \Rightarrow \int_{-1}^{1} f(\xi) \, d\xi = \int_{-1}^{1} 1 \, d\xi = 2 = w_1 \cdot f(\xi_1) = w_1 \Rightarrow w_1 = 2 \]

2nd defining equation: 1st order polynomial \( f(\xi) = \xi \)

\[ \Rightarrow \int_{-1}^{1} f(\xi) \, d\xi = \int_{-1}^{1} \xi \, d\xi = 0 = w_1 \cdot f(\xi_1) = w_1 \xi_1 \Rightarrow \xi_1 = 0 \]

Thus, the 1st Gauss formula is:

\[ \int_{-1}^{1} f(\xi) \, d\xi = 2 \cdot f(0) \]

D.P.1 This formula is exact up to the polynomial order \( n = 1 \)

Example: \( f(\xi) = 3 + \xi \) \( \Rightarrow \int_{-1}^{1} f(\xi) \, d\xi = (3 + \xi) \left( \frac{\xi^2}{2} \right)_{-1}^{1} = 6 \) \( \Rightarrow \) exact

Gauss formula \( \int_{-1}^{1} f(\xi) \, d\xi = w_1 f(\xi_1) = 2 \cdot f(0) = 2 \cdot 3 = 6 \)

D.P.2 1st Gauss formula \( \Leftrightarrow \) Mid-point Integration rule.
Gauss Quadratures

The integration formulas (Newton-Cotes) were of the form

\[ \int_a^b f(x) \, dx = \omega_1 f(x_1) + \omega_2 f(x_2) + \ldots + \omega_n f(x_n) \]

where sample points were selected in advance and weights are pertinent to the method itself.

* These formulas are exact for polynomials of order up to \(N-1\).

1. How can we determine the sample points \(x_k\) and weights \(w_k\)?
   - This will increase the order of accuracy.

2. What if both sample points \(x_k\) and weights \(w_k\) are variables?
   - Gauss! Also known as abscissas.

\(N=1\):

One point Gauss Integration

\[ I = \int_a^b f(x) \, dx = \omega_1 f(x_1) \Rightarrow \text{unknown} \quad \omega_1, \quad \text{need 2 equations} \]

1st defining equation: \(0^{th}\) order polynomial \(f(x) = 1\)

\[ I = \omega_1 = \int_a^b 1 \, dx = 2 \Rightarrow [\omega_1 = 2] \]

2nd defining equation: \(1^{st}\) order polynomial \(f(x) = x\)

\[ I = \int_a^b x \, dx = \frac{x^2}{2} \bigg|_{-1}^{1} = 0 = \omega_1 f(x_1) = 2f(x_1) \Rightarrow f(x_1) = 0 \Rightarrow [x_1 = 0] \]

Thus, 1st Gauss formula is:

\[ \int_a^b f(x) \, dx = 2f(0) \]

Discussion points:

- This formula is exact up to polynomial order \(N = 1\).
- 1st Gauss Quadrature \(\equiv\) Mid-point rule.
2 \ n=2 \ Two-point \ Gauss \ Quadrature

\[ \int_{-1}^{1} f(x) \, dx = w_1 f(x_1) + w_2 f(x_2) \]

- need to find 4 quantities: \( w_1, w_2, x_1, x_2 \) \( \in [-1, 1] \)
- need to form 4 equations:

\[ E: \text{monomials of order } x^0, x^1, x^2, x^3 \text{ need to be used} \]

\[ \text{If solution exist it will be exact up to the } 3^{rd} \text{ order polynomial.} \]

**Defining Equations:**

1. \( f(x) = 1 \rightarrow \int_{-1}^{1} f(x) \, dx = \int_{-1}^{1} 1 \, dx = 2 = w_1 + w_2 \)

2. \( f(x) = x \rightarrow \int_{-1}^{1} f(x) \, dx = \int_{-1}^{1} x \, dx = 0 = w_1 x_1 + w_2 x_2 \)

3. \( f(x) = x^2 \rightarrow \int_{-1}^{1} f(x) \, dx = \int_{-1}^{1} x^2 \, dx = \frac{2}{3} = w_1 x_1^2 + w_2 x_2^2 \)

4. \( f(x) = x^3 \rightarrow \int_{-1}^{1} f(x) \, dx = \int_{-1}^{1} x^3 \, dx = 0 = w_1 x_1^3 + w_2 x_2^3 \)

Thus, we have a system of 4 equations with 4 unknowns.

The solution of this system yields: \( w_1 = w_2 = 1, x_1 = -\frac{1}{\sqrt{3}}, x_2 = \frac{1}{\sqrt{3}} \)

Thus, the 2nd Gauss Quadrature is:

\[ \int_{-1}^{1} f(x) \, dx = f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}) \]

**Discussion Points**

1. What do we do for \( \int_{a}^{b} f(x) \, dx \)?

   Assume \( x = \frac{b-a}{2} t + \frac{a+b}{2} \Rightarrow dx = \frac{b-a}{2} \, dt; x|_{a}^{b} \rightarrow 0, 1; f(x) \rightarrow g(t) \)

   i.e., \( \int_{a}^{b} f(x) \, dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2} t + \frac{a+b}{2}\right) \, dt = \frac{b-a}{2} \int_{-1}^{1} g(t) \, dt \)

   and \( m^{th} \) point Gauss Quadrature is:

\[ \int_{a}^{b} f(x) \, dx = \frac{b-a}{2} \sum_{k=1}^{m} w_k \, g(t_k) \]
It can be shown that:
\[
\int_a^b f(x)\,dx - Q_{am} = \frac{(b-a)^{m+1}}{(2m+1)!\,[2^m\cdot(2m)!]} M_2 m
\]
where \( M_2 m \) is a constant such that \( \int_{-\infty}^{\infty} f(x)^2 \,dx \leq M_2 m \).

**Example:** Use Eq. 6.2. formula to compute \( f(x) = x^5 + \frac{1}{x} x^2 - 3x \)
on \( x \in [-1, 1] \) and find an absolute error.

**STEP:** draw the function to make sure no singularities.
- Analytica rule: \( n = 116 \)
- \( Q = \frac{1}{2}\int_{-1}^{1} f(x)\,dx = \frac{1}{2}(f(1) + f(-1)) = \frac{1}{2}(1 + (-1)) = 0 \)

=> Absolute error is 0.

**Def:** A system of functions \( \{\phi_n\} (n = 0, 1, \ldots) \) is said to be orthogonal with weight \( w(x) \) on the interval \((a,b)\) if:
\[
\int_{a}^{b} w(x) \phi_n(x) \phi_m(x) \,dx = 0 \quad \text{for} \quad m \neq n
\]

For \( \int f(x) \,dx \) use Legendre polynomials: these are also referred to as Gaussian quadratures: \( x_i , w_i \) are tabulated.

**Def:** If function \( f(x) \) is defined \( \forall x \in [0,1] \), then use Gauss-Laguerre's quadratures:
\[
\int_{0}^{1} f(x)\,dx = \sum_{k=1}^{m} w_k f(x_k) \quad \text{use approximate table.}
\]

**Def:** If function \( f(x) \) is defined \( \forall x \in (-\infty,0) \), then use Gauss-Hermitean quadratures:
\[
\int_{-\infty}^{0} f(x)\,dx = \sum_{k=1}^{m} w_k f(x_k) \quad \text{use approximate table.}
\]
**Convergence of numerical integration**

1) **Which factors determine the accuracy of a numerical method?**
   - Resolution: number of grid points per interval (wavelength)
   - Order of polynomial expansion/quadrature (low/higher-order b.f.)
   - Sampling (mesh) quality

   - If a numerical (CEM) method works properly, then for an infinitely accurate machine, the increased resolution/sampling will always improve accuracy.
   - However, mostly the error is not zero.
   - To improve the accuracy we have to:

<table>
<thead>
<tr>
<th>Increase Resolution (h-refinement)</th>
<th>Increase Polynomial Order (p-refinement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>In some selected computational domain</td>
</tr>
<tr>
<td>Non-uniform</td>
<td>Parts of domain</td>
</tr>
<tr>
<td>Easy but not</td>
<td>Throughout the</td>
</tr>
<tr>
<td>Very efficient</td>
<td>In some selected computational domain</td>
</tr>
<tr>
<td>Doubly</td>
<td></td>
</tr>
</tbody>
</table>

**CEM Example:**

1. Inhomogeneous intrusion of materials with different properties
2. Structured mesh $\rightarrow$ Small + Large details in domain.
3. Inlet problem $\rightarrow$ Around the feed port.

In this course we consider mainly uniform refinement.

2) **How do we determine the error?**
   - Different error measures exist: absolute vs. relative.
   - They all imply that the exact solution is known a priori, and that the error is obtained in an a posteriori way.
   - In CEM we use term convergence of a method to determine how good is a method in obtaining the solution.
   - For a specific problem the convergence of a method (FD, FEM) can be difficult to determine.
Example
Calculate electrostatic potential on the symmetry axis of a uniformly charged square for \( z = z_0 \) any midpoint & Simpson's rules and establish their order of convergence.

**STEP 1**

Formulae:

Given:
1. Square \( z = z_0 \), \( a, b \), \( y(x) \)?
2. Uniformly charged: \( \phi(x, y, z) = \text{const.} \)

Solve:
\[ \phi(0, 0, a) = ? \]

(a) What is the potential of a single charge?
\[ \phi = \frac{Q}{4\pi \epsilon_0 r} \Rightarrow \Delta \phi = \frac{\Delta Q}{4\pi \epsilon_0 r} \]

Thus
\[ \Delta \phi = \frac{Q}{4\pi \epsilon_0 r} \text{ and } \phi = \frac{Q}{4\pi \epsilon_0} \int \frac{ds}{r} \]

For \( \phi(0, 0, z) = \frac{Q}{4\pi \epsilon_0} \int dx \int \frac{dy'}{\sqrt{x^2 + y'^2 + z^2}} \]

At \( z = z_0 \) we need to compute
\[ I = \frac{Q}{4\pi \epsilon_0} \int dx \int \frac{dy'}{\sqrt{x^2 + y'^2 + z_0^2}} \]

**STEP 2**

Solution:
We can do it analytically, however, we will use midpoint and Simpson's rules.

Midpoint:
\[ \int f(x) dx = h \cdot f(x + \frac{h}{2}) \]

Simpson's:
\[ \int f(x) dx = \frac{h}{3} \left( f(x) + 4f(x + \frac{h}{2}) + f(x + h) \right) \]

For \( z = z_0 = 1 \) we obtain the following:

<table>
<thead>
<tr>
<th>( h )</th>
<th>( n )</th>
<th>( \int )</th>
<th>Correct answer up to 8 digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.79452 30171</td>
<td>0.79335 94378</td>
</tr>
<tr>
<td>7</td>
<td>0.14286 x 2</td>
<td>0.79385 04952</td>
<td>0.79335 92042</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.79357 97873</td>
<td>0.79335 91413</td>
</tr>
<tr>
<td>15</td>
<td>0.0667 x 2</td>
<td>0.79346 60584</td>
<td>0.79335 91252</td>
</tr>
<tr>
<td>20</td>
<td>0.08</td>
<td>0.79341 92684</td>
<td>0.79335 91225</td>
</tr>
</tbody>
</table>
**STEP 1** Carry out integrations for various samplings (mesh sizes) \( H, (h) \) and store data in respective vectors.
\[ \mathbf{H} = [h, h, h, \ldots, h_n] \]
\[ \mathbf{I}_{\text{mp}} = [I_{m}, I_{m\over 2}, I_{m}, \ldots, I_{m\over 2\over 3}] \]
\[ \mathbf{I}_{S} = [I_{S}, I_{S\over 2}, I_{S}, \ldots, I_{S\over 2\over 3}] \]

**STEP 2** Create vectors with powers of sampling:
\[ \mathbf{H}, \mathbf{H}^2, \mathbf{H}^3, \ldots, \mathbf{H}^k \]

**STEP 3** Plot functions \((\mathbf{H}^k, \mathbf{I}_{\text{mp}})\) \(k = 1, 2, 3\), this "k" establishes order of convergence.

For our case \(k=2\) would give linear function.

**STEP 4** Recognized \(k\) for which plot \((\mathbf{H}^k, \mathbf{I}_{\text{mp}})\) is linear.

- Alternative: Plot \(\mathbf{I}_{\text{mp}}\) vs \(\log \mathbf{H}\) until pet linear curve

- Note: Another simple way to establish \(H\)-convergence is to carry out the computations for geometric series of mesh sizes such as \(h, h/2, h/4, \ldots\). The ratio \(h/t\) for \(h = \{0.4, 0.2, 0.1\}\) gives \(2.002\) midpoint and \(3.985\) Simpson's.
Q) Why do we care about matrices?

As a result of Formulation, many EM BVPs end up being represented by the system of linear algebraic equations:

\[
\begin{align*}
\begin{cases}
  a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\
nx_2 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\
\vdots \\
nx_n + a_{nn}x_n + \ldots + a_{nn}x_n = b_n
\end{cases}
\end{align*}
\]

or \[A\mathbf{x} = \mathbf{b} \]

\[A_{nn} \text{ Square matrix} \quad \text{coefficient matrix filled by } \text{Helm, FEM, etc.}\]

\[\mathbf{x} : \text{unknown vector} \quad \mathbf{b} : \text{excitation vector}\]

To find the solution for our BVP we have to solve \[A^{-1}\mathbf{b} = \mathbf{x}\]

Thus, for efficient solution of the problem, we must have a well understanding of matrix \(A\).

Q) Which matrix properties are important?

1. Matrix \(A = [a_{ij}]_{n \times n}\) is real if \(a_{ij} \in \mathbb{R}\)
2. Matrix \(A = [a_{ij}]_{n \times n}\) is complex if \(a_{ij} \in \mathbb{C}\)
3. Matrix \(A = [a_{ij}]_{n \times n}\) is symmetric if \(A = A^T\) or \(a_{ij} = a_{ji}\)
4. Matrix \(A = [a_{ij}]_{n \times n}\) is Hermitian: \(A = (A^*)^T\) if \(a_{ij} = \overline{a_{ji}}\)
5. Matrix \(A = [a_{ij}]_{n \times n}\) is Toeplitz when \(a_{ij}\) depends only on \(|i-j|\)
or when elements are constant on each diagonal
\[
\begin{bmatrix}
a_0 & a_1 & \cdots & a_n \\
a_1 & a_0 & \cdots & a_{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}
\]
6. Matrix \(A = [a_{ij}]_{n \times n}\) is circulant: subclass of Toeplitz
\[
\begin{bmatrix}
a_0 & a_1 & \cdots & a_{n-1} \\
a_1 & a_0 & \cdots & a_{n-2} \\
\vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}
\]
7. Matrix \(A = [a_{ij}]_{n \times n}\) is positive definite if \(\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}, \mathbf{x}^T A \mathbf{x} > 0\)

(Important for Cholesky decomposition: \(A = LL^T\))

8. Matrix norm \(\|A\|_{\infty}\) is a function \(f: \mathbb{R}^{n \times n} \rightarrow \mathbb{R}\) such that:
\[
\begin{align*}
f(A) & > 0 \quad A \in \mathbb{R}^{n \times n} \quad (f(A) = 0 \Rightarrow A = 0) \\
f(cA) & = f(A) \quad c \in \mathbb{R} \quad (A, B \in \mathbb{R}^{n \times n}) \\
f(A) & \leq f(A + B) \quad f(A) \leq f(B) \quad A, B \in \mathbb{R}^{n \times n} \\
f(A \cdot B) & \leq f(A) \cdot f(B) \quad A, B \in \mathbb{R}^{n \times n}
\end{align*}
\]
notation: \( \| A \|_p = \text{norm}(A, p) \)

Different norms exist (they are all equivalent):
\[
\| A \|_F = \sqrt{\sum \sum |a_{ij}|^2} \quad \text{Probenius:} \quad \text{norm}(A, \text{Frob})
\]
\[
\| A \|_\infty = \max_i \sum_j |a_{ij}| \quad \text{norm}(A, \infty)
\]

**Discussion Points**

**DP1.** Matrix is sparse: for PDE methods

**DP2.** Matrix is dense: for IE methods

**ANXX:** \( \text{N-number of unknowns} \leq E, H, \gamma \) for FEM

**DP3.** For systems where \( N \approx 10^3 \), a number of efficient algorithms exist for solving \( Ax = b \).

However, many problems result in \( N \approx 10^6 \) unknowns and knowing some properties of \( A \) can be very helpful in those cases.

**DP4.** What are practical constraints in solving \( Ax = b \) on computers?

1) memory - more critical than!
2) execution (running) time - important but not as severe.

**Memory constraints might preclude solution:**

- **Ex.:** A fighter jet at X-band results in about \( N \approx 5 \times 10^6 \) unknowns.

Thus we have to fill \( N^2 = 25 \times 10^{12} \) matrix entries, i.e., if each entry is single precision (complex number) demanding 2 x 4 bytes, then we need to reserve \( 2 \times 4 \times (5 \times 10^6)^2 \times 2 \approx 200 \text{TB} \) of RAM.

Thus we should know some specific properties of matrices to reduce computational burden.

**DP5.** If a system (coefficient) matrix is nearly singular, then the solution we can obtain might be incorrect.

To quantify the level of "near-singularity," we can calculate matrix norms and specifically the condition.
Condition number and error propagation

Two sources of errors are distinct in solving LSEs:
1) Errors in computing matrix and vector entries $a_{ij}, b_i$:
   - utilized approximations
   - truncation due to machine finite precision

2) Errors in inverting $A^{-1}$ and solving $x = A^{-1}b$ due to
   - round-off and truncations

To predict the impact of these and to understand them, we investigate the **condition number** for our system matrix $A$.

If $\|A\|, \|A^{-1}\| \leq p$, the condition number is defined as:

$$\kappa_p(A) = \|A\|_p \cdot \|A^{-1}\|_p$$

In Matlab:

$$\text{cond}(A) = \text{norm}(A, 2) \cdot \text{norm(inv(A), 2})$$

If we know the eigenvalues of $A$, we can estimate the condition number as:

$$\kappa_p(A) \leq \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

**Examples:**

1. Consider the following linear system $Ax = b$:

   $$A = \begin{bmatrix} 0.78 & 0.543 \\ 0.913 & 0.652 \end{bmatrix}, \; b = \begin{bmatrix} 0.341 \\ -0.087 \end{bmatrix}$$

   After applying two different methods, we obtained solutions:

   $$x^{(1)} = \begin{bmatrix} 0.989 \\ -1 \end{bmatrix}$$

   and

   $$x^{(2)} = \begin{bmatrix} 0.999 \\ -1 \end{bmatrix}$$

   Which solution is more accurate?

   - Obvious way is to compute the residuals:
The exact solution is $x_{\text{exact}} = [1 \ 1]^T$. Is it?

Conclusions from example 1:

1. Notion of a good solution is ambiguous.
2. We need to understand mathematics better behind $Ax=b$.
3. Matrix $A$ is nearly singular.
   Specifically, matrix $A = \begin{bmatrix} 0.78 & 0.563 \ 0.913 & 0.659 \end{bmatrix}$ is singular.

The condition number of $A$ is $\kappa_2(A) = 2.19 \times 10^6$ thus small perturbation in $A$ or $b$ can create great change in solution vector.

Given $A = \begin{bmatrix} 1 & 1.01 \ 1 & 2.01 \end{bmatrix}$, $b = \begin{bmatrix} 2 \ 2.01 \end{bmatrix} \Rightarrow x_{\text{exact}} = \begin{bmatrix} 1 \\
1 \end{bmatrix}$.

If we change $b_2$ by only 0.5% to 2.02 our exact solution is now $x_{\text{exact}} = \begin{bmatrix} 2 \\
2 \end{bmatrix}$, a change for 100%.

The condition number of $A$ is:

$\kappa_2(A) = \frac{\|A\| \|A^{-1}\|}{\|A^{-1}\|} = 404 \ (p=1)$
$\kappa_2(A) = \frac{\|A\| \|A^{-1}\|}{\|A^{-1}\|} = 402 \ (p=2)$
$\kappa_2(A) = \frac{\|A\| \|A^{-1}\|}{\|A^{-1}\|} = 404 \ (p=1)$

Thus, we want $\kappa_2(A)$ to be as close to 1 as possible.

Q2 Why is this happening?

Let's consider perturbation equation: $(A+\delta A)(x+\delta x) = b+\delta b \Rightarrow \delta x = A^{-1} \delta b$ and

$\delta x \rightarrow$ perturbation in solution
$\delta b \rightarrow$ perturbation in excitation

From $Ax = b \Rightarrow \|\delta x\| \leq \|A\| \cdot \|\delta b\|$. If $\|\delta b\| \approx 0$ then $\|\delta x\| \approx 0$

Thus $\frac{\|\delta x\|}{\|x\|} \leq \|A^{-1}\| \cdot \|\delta b\| = \kappa_2(A) \cdot \|\delta b\| = \frac{\|\delta b\|}{\|b\|}$

$\kappa_2(A)$ large $\Rightarrow$ perturbation is huge.
Algorithms for the solution of LSEs

\[ \text{Let } b = y \Rightarrow \text{Solve } \mathbf{A} \mathbf{x} = \mathbf{b} \]

To find \( \mathbf{x} \) we need to solve \( \mathbf{x} = \mathbf{A}^{-1} \mathbf{b} \)

2. How do we solve \( \mathbf{A} \mathbf{x} = \mathbf{b} \)?

- **Direct Methods** → Gaussian Elimination, LU decomposition, Cholesky decomposition, QR factorization
- **Iterative Methods** → Jacobi, Gauss-Seidel, gradient, QMR, GMRES, etc.

**Naive Gaussian Elimination**

- **Simplest**

\[
\begin{bmatrix}
 a_{11} & a_{12} & \cdots & a_{1n} \\
 a_{21} & a_{22} & \cdots & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
 x_1 \\
 x_2 \\
 \vdots \\
 x_n
\end{bmatrix} =
\begin{bmatrix}
 b_1 \\
 b_2 \\
 \vdots \\
 b_n
\end{bmatrix}
\]

and then use back substitution to find \( \mathbf{x} \)

Start with augmented form:

\[
\begin{bmatrix}
 a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\
 a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 a_{n1} & a_{n2} & \cdots & a_{nn} & b_n
\end{bmatrix}
\begin{bmatrix}
 x_1 \\
 x_2 \\
 \vdots \\
 x_n
\end{bmatrix} =
\begin{bmatrix}
 b_1 \\
 b_2 \\
 \vdots \\
 b_n
\end{bmatrix}
\]

**Notes**:

1. Total number of additions & multiplications \( \mathbf{A} \) : \( O\left(\frac{N^3}{3}\right) \)

2. Not efficient approach when multiple RHSs are given (scattered from different directions) \( \Rightarrow O(N\frac{N^2}{3}) \) for each RHS.
After $N$ steps we end up with

\[
\begin{bmatrix}
1 & a_{21}^{(1)} & a_{31}^{(1)} & \cdots & a_{N1}^{(1)} & b_{1}^{(1)} \\
1 & a_{22}^{(2)} & a_{32}^{(2)} & \cdots & a_{N2}^{(2)} & b_{2}^{(2)} \\
1 & a_{23}^{(3)} & a_{33}^{(3)} & \cdots & a_{N3}^{(3)} & b_{3}^{(3)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & a_{2N}^{(N-1)} & a_{3N}^{(N-1)} & \cdots & a_{NN}^{(N-1)} & b_{N}^{(N-1)}
\end{bmatrix}
\begin{bmatrix}
x_1^{(1)} \\
x_2^{(2)} \\
x_3^{(3)} \\
\vdots \\
x_{N-1}^{(N-1)}
\end{bmatrix}
= \begin{bmatrix} b_1^{(1)} \\
b_2^{(2)} \\
b_3^{(3)} \\
\vdots \\
b_{N-1}^{(N-1)}
\end{bmatrix}
\]

Discussion Points:

1. Total number of multiplications and additions is $O\left(\frac{N^3}{3}\right)$ or
   exactly $\frac{1}{3}N^3 + \frac{1}{2}N^2 - \frac{5}{6}N$

2. Major drawback: What if we have multiple right-hand sides? Ex: scattering when excitation comes from various incidence angles!!!

- In this case, $b$ has to be computed for each incidence and GE must be performed for each new excitation.
- This will produce $O\left(\frac{N^3}{3}\right)$ effort for each incidence.

3. Factorize $A$ as $A = LU$ where $L$ - lower triangular and $U$ - upper triangular matrix of the form:

\[
L = \begin{bmatrix}
1 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
1 & 1 & 0 & \cdots & 0 \\
0 & 1 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1
\end{bmatrix}
= \begin{bmatrix}
an & an & \cdots & an \\
an & an & \cdots & an \\
an & an & \cdots & an \\
\vdots & \vdots & \ddots & \vdots \\
an & an & \cdots & an
\end{bmatrix}
\]

$L$ $U$ can be obtained with the same computational cost as a simple GE and they can be stored in the memory method of $A$. 
Thus, we perform LU decomposition once, and perform backward and forward substitutions for each new vector \( \mathbf{b} \).

\[
\mathbf{A} \mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{LU} \mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \mathbf{L} \mathbf{c} = \mathbf{b} \quad \Rightarrow \quad \text{forward substitution}
\]

\[
\mathbf{U} \mathbf{z} = \mathbf{c} \quad \Rightarrow \quad \text{find } \mathbf{z} \quad \text{by backward substitution}
\]

Steps:
1. Decompose \( \mathbf{A} = \mathbf{LU} \)
2. Solve \( \mathbf{L} \mathbf{c} = \mathbf{b} \) -> find \( \mathbf{c} \) by forward substitution
3. Solve \( \mathbf{U} \mathbf{z} = \mathbf{c} \) -> find \( \mathbf{z} \) by backward substitution

* Order of multiplications and additions:

\[
\text{Order of multiplications and additions:}
\]

Thus, instead of having \( \mathcal{O}(n^3) \) for each new \( \mathbf{b} \), we have it only once and then for each \( \mathbf{b} \) we have \( \mathcal{O}(n^2) \)

* For large \( n \), factorization accounts for a bulk of time

* (Q) What happens if diagonal elements \( a_{ii} = 0 \)?
  - Algorithm fails
  - Use pivoting: replace the row with the one which has a diagonal element with new row that doesn't have 0 at that spot
  - Different pivoting algorithms exist. The best search for the largest element in that column to move at that 0 point.
  - This pivoting also reduces numerical error.

(4) If matrix \( \mathbf{A} \) is symmetric p.d. then there exist a unique lower triangular matrix \( \mathbf{L} \) such that \( \mathbf{A} = \mathbf{LL}^T \)

\[
\mathbf{A} = a_{ij} = a_{ji} \quad \text{p.d.} \quad \forall x, y \in \mathbb{R} \quad x^T \mathbf{A} x \geq 0
\]

\[
\mathbf{L} : a_{ij} = 0 \quad \forall i > j
\]

- This is Cholesky decomposition:

  - Order is \( \mathcal{O}(n^{3/6}) \)
  - Due to symmetry, need to compute only \( \frac{1}{2} \) of \( \mathbf{A} \)
Iterative Methods:

1) Why?
- Computational cost of a direct solver is too high once \( N > 10^4 \).
- \( \Theta(N^3) \) operations, \( \Theta(N^2) \) storage.
- For large \( N \): ill-conditioned system matrix and round-off errors can produce some problems (inaccurate results).

2) Iterative methods allow for:
   a) Reduced RUNTIME
   b) Improved accuracy due to reduced round-off errors
   c) Some methods allow for reduced memory storage

3) Which methods are used in CEM?

Iterative Methods:

- Stationary
  - Jacobi
  - Gauss-Seidel
  - Relaxation
- Non-stationary
  - Gradient
  - Conjugate gradient
  - Conjugate symmetric
  - Quasi-minimum residual (QMR)
  - General minimum residual (GMRES)

Jacobi Method:

Given non-singular coefficient matrix \( A = a_{ij} \): \( a_{ii} \neq 0 \), \( i = 1, 2, \ldots, N \)
form the system of equations:

\[
\begin{align*}
a_{11} x_1 + a_{12} x_2 + \ldots + a_{1N} x_N &= b_1 \\
a_{21} x_1 + a_{22} x_2 + \ldots + a_{2N} x_N &= b_2 \\
\vdots \\
a_{N1} x_1 + a_{N2} x_2 + \ldots + a_{NN} x_N &= b_N
\end{align*}
\]

Find \( x = [x_1, \ldots, x_N]^T \).
**STEP 0**  
\[ k = 0 : \]
\[
x_1 = \frac{1}{a_{11}} \left[ b_1 - (a_{12} x_2 + a_{13} x_3 + \ldots + a_{1n} x_n) \right]
\]
\[
x_n = \frac{1}{a_{nn}} \left[ b_n - (a_{n1} x_1 + \ldots + a_{nn} x_n) \right]
\]

**STEP 1**  
At the current guess \( x_i^{(0)} = 0 \), i = 1, \ldots, n
\[
x_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k-1)} \right] = b_i / a_{ii}
\]
\[
x_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k-1)} \right] = b_i / a_{ii}
\]

**STEP 2**  
Check if the residual is smaller than a given tolerance:
\[ r^{(k)} = A x^{(k)} - b \]

- If \( \| r^{(k)} \| \leq \text{tol} \) then go to \( P_0 \) and set \( k = k + 1 \)
- If \( \| r^{(k)} \| > \text{tol} \) then update \( x_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k-1)} \right] \) and go back to \( \text{STEP 1} \)

**END**
Discussion Points:

1. Easiest method but convergence is slow

2. Criteria for stopping iterations can be different:

   \[ \frac{\|X^{(k)} - X^{(k-1)}\|}{\|X^{(k)}\|} \leq \text{tol} \]

3. Note that when \( X_i^{(k)} \) is computed we already know \( X_{i-1}^{(k)} \ldots X_1^{(k)} \) thus, we can change the algorithm to:

   \[ X_i^{(k)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1}^{n} a_{ij} X_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} X_j^{(k-1)} \right] \]

   Gauss-Seidel method

   \[ \Rightarrow \text{converges 2 times faster} \]

4. Non-stationary methods:
   - much faster convergence
   - more complex
   - common algorithm:
     - Start with initial guess for \( X \rightarrow X^{(0)} \)
     - Compute residual vector \( \Rightarrow \Gamma_i^{(k)} = A \cdot X^{(k)} - b \)
     - Use residual vector to correct current solution vector before repeating for iteration:
       \[ X^{(k+1)} = X^{(k)} + \alpha_{k+1} P_{k+1} \] where \( \alpha_{k+1} \in \mathbb{R}^{n \times 1} \)

   \( \Gamma \) both are function of \( \alpha \), residual

   - Iteration stop when normalized residual error is less than tolerance:

   \[ \frac{\|\Gamma^{(k)}\|}{\|b\|} \leq \text{tol} \]
What can we do to improve convergence?

Preconditioning

- If coefficient matrix $A$ is ill-conditioned, we should use a pre-conditioner.

**Motivation:**

If $Ax = b \Rightarrow \text{find } P \in \mathbb{C}^{n \times n} : PAx = Pb$ and

\[ \text{cond } (P^{-1}A) \ll \text{cond } (A) \]

- Obvious choice: $P = A$ or $P \approx A$

- The nonsingular matrix $P$ has to satisfy the following:
  - Should be a good approximation of $A$
  - Should be easy to compute
  - Should be invertible in $O(n)$ operations

**Example:**

The simplest pre-conditioner is point diagonal for which

\[ P_{ij} = \delta_{ij} A_{ij} \quad i = 1, 2, \ldots, n \rightarrow \text{takes only diagonal elements of } A \text{ to have pre-conditioner } P \]

If $A = \begin{bmatrix} 1 & 1 \\ 0 & 10^4 \end{bmatrix}$ find pre-conditioner matrix $P$ : cond($P^{-1}A$) < cond($A$)

\[ \text{cond } (A) \approx 10^4 \]

Take diagonal pre-conditioner $P = \begin{bmatrix} 1 & 0 \\ 0 & 10^4 \end{bmatrix} \Rightarrow P^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-4} \end{bmatrix}$ and now $P^{-1}A = \begin{bmatrix} 1 & 1 \\ 0 & 10^{-4} \end{bmatrix}$ and cond($P^{-1}A$) = 2.62

**Note:** We have to be very careful with this pre-conditioner.

- If $A_{11} = 10 \Rightarrow \text{cond } (P^{-1}A) = 102 < \text{cond } (A) = 10^4$
- If $A_{11} = 100 \Rightarrow \text{cond } (P^{-1}A) = 10^4 \approx \text{cond } (A) = 10^4$
- If $A_{11} = 1000 \Rightarrow \text{cond } (P^{-1}A) = 10^6 > \text{cond } (A) = 10^4$
Finite Difference Method 2-7-05  (Chapter 3)

Outline
Introduction
FD schemes

Q) What is a FO method (FDM)?

2/ FDM is a PDE based method for solving (primarily) static HE's when used in EM. It can also be used for eigenvalue problems.

FDM is based on numerical approximations of derivatives by finite differences

FDM thus also belongs to the class of fridging methods.

FDM can be derived from MOM.

FDM is typically used for the closed domain problems, and quasi-static analysis of transmission lines (con, microstrip...)

FDM was developed in 1920's for solving non-linear equations arising in hydrodynamics

FDM is simplest of all methods.

FDM involves 4 steps (once we know the governing equation and related boundary conditions)

STEP 0 Derivatives of governing equations and b.c.'s

L \phi = g and b.c. on \partial \Omega
**STEP 1**  Discretization / Meshing / Gridding

- A nodal grid is established within the computational domain.

In this step, we also should decide:

- Boundary nodes \( \rightarrow \) nodes on \( \partial \Omega \)
- All other nodes where some boundary conditions apply
- Nodes where potentials/fields are to be computed

**Common gridding schemes:**

- Rectangular
- Skewed
- Triangular
- Circular

**STEP 2** Setup FD equations

2.1) Form approximations for the field/potential derivatives that relate values of the fields/potentials at each node with the values at neighboring nodes

2.2) Cast system of LSEs \( A x = b \) a set of eq. 2.1

**STEP 3** Solve the LSE

- Find \( x = A^{-1} b \) \( \rightarrow \) direct or iterative methods

**STEP 4** Extraction of Parameters

- Compute required parameters from fields/potential values
  - Capacitance, charge, phase velocity, characteristic impedance, eigenvalues, ...
(2) How do we derive finite differences?

Given an analytic function \( f(x) \) over an interval \([a, b]\), determine its derivative at point \( P \).

Find \( f'(x_0) \).

Denote nodes:

\[ A : f(x_0 - \Delta x) \]
\[ B : f(x_0 + \Delta x) \]
\( \Delta x \) arbitrarily small value.

The \( f'(x_0) \) is the slope of the tangent at \( P \) wrt \( x \) axis \( x \).

We can approximate this slope (tangents) as:

- **Forward difference**
  \[ \tan \theta (PB) : \quad f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \]

- **Backward difference**
  \[ \tan \theta (AP) : \quad f'(x_0) \approx \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} \]

- **Central difference**
  \[ \tan \theta (AB) : \quad f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} \]

Thus any approximation in terms of values at discrete points is called F.D. approximation.

To obtain

(2) How accurate are these approximations?

To obtain mathematical foundation for the error estimate, i.e., how good is the approximation, we recall that an analytic function \( f(x) \) can be developed in Taylor series as:

\[ f(x + \Delta x) = f(x_0) + \Delta x \cdot f'(x_0) + \frac{\Delta x^2}{2} f''(x_0) + \frac{\Delta x^3}{3!} f'''(x_0) + \cdots \]

In A:
\[ f(x_0 - \Delta x) = f(x_0) - \Delta x \cdot f'(x_0) + \frac{\Delta x^2}{2} f''(x_0) - \frac{\Delta x^3}{3!} f'''(x_0) + \cdots \]

Then we have:
forward difference: \( f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + O(\Delta x) \)

backward difference: \( f'(x_0) \approx \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} + O(\Delta x) \)

central difference: \( f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} + O(\Delta x^2) \)

Notes/Observations

(1) central difference approximation is more accurate than both forward and backward differences

(2) error is introduced since we truncate to Taylor series

(3) higher-order approximations can be obtained by using more terms in Taylor series

\[
\text{Forward Difference: } f'(x_0) \approx \frac{f(x_0 + 2\Delta x) + 4f(x_0 + \Delta x) - 3f(x_0)}{2\Delta x} + O(\Delta x^2)
\]

\[
\text{Central Difference: } f'(x_0) \approx \frac{f(x_0 + 2\Delta x) + 8f(x_0 + \Delta x) - 8f(x_0 - \Delta x) + f(x_0 - 2\Delta x)}{12\Delta x} + O(\Delta x^4)
\]

(4) 2nd order derivatives finite difference approximations can be obtained by 3rd order differences

low order \( f''(x_0) = \left(f'(x_0)\right)' = \frac{f(x_0 + 2\Delta x) - 2f(x_0) + f(x_0 - 2\Delta x)}{4\Delta x} + O(\Delta x^4) \) central difference

and higher order \( f''(x_0) \approx -\frac{f(x_0 + 2\Delta x) + 16f(x_0 + 2\Delta x) - 17f(x_0) + 16f(x_0 - 2\Delta x) - f(x_0 - 2\Delta x)}{8\Delta x^2} + O(\Delta x^4) \)