Iterative Methods

- used for large linear systems.

- used for nonlinear systems even for a single equation.

- Here we have an iterative Error because we stop the process when we are close enough to the exact solution.

## Types of Errors:

1. Modeling Error. 
   \[ s" = \text{curvature} \]
   because \[ \text{curvature} = \frac{s"}{(1 + s'^2)^{3/2}} \]

2. Round-off Error. 
   "due to limited memory of computers."

3. Iterative Error. 
   \[ \bar{X}_{\text{sol}} = \bar{X}_{\text{Exact}} + \bar{X}_{\text{Iter.}} \]

4. Truncation Error. 
   \[ \frac{dx}{dt} = \frac{X_{n+1} - X_i}{t_{n+1} - t_i} + O(h) \]
   \[ \text{Truncation Error} \]
* Using iterative methods introduce stability problems which is minimized if the system matrix is diagonally dominant, i.e.

\[ |a_{i,i}| > \sum_{j \neq i} |a_{i,j}| \]

* At least for one row \( a_{i,i} > \sum_{j} a_{i,j} \)

* Example: Consider the tridiagonal system

\[ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \]

\( \Rightarrow \) Performing Gaussian elimination will lead to

\[ b_i' = b_i - a_i \frac{c_{i-1}}{b_{i-1}} \]

If the matrix was diagonally dominant then

\[ \left| \frac{c_{i-1}}{b_{i-1}} \right| < 1 \quad \Rightarrow \quad b_i' \leq 0 \]

\( \Rightarrow \) No stability problem where \( |b_i| > |a_i| \)

Note that we have stability problem if \( \frac{c_{i-1}}{b_{i-1}} = 0 \)
1. Jacobi Method:

\[ \mathbf{A} \mathbf{x} = \mathbf{b} \implies \mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} \]

where \( b_i^{(n)} = b_i - \sum_{j \neq i} a_{i,j} x_j^{(n)} \)

⇒ Procedure:

Step 1: Make a guess of all unknowns

Step 2: Solve the simplified system \( \mathbf{x}^{(n+1)} \)

Step 3: Calculate the Residue

\[ \text{Res} = \| \mathbf{A} \mathbf{x} - \mathbf{b} \| \]

Step 4: If \( \max(\text{Res}) \leq \text{tolerance} \) then you are done, else use \( \mathbf{x}^{(n+1)} \) instead of your initial guess & repeat.

* for tridiagonal systems:

\[ a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \]

new \( x_i = \frac{d_i - a_i x_{i-1} - c_i x_{i+1}}{b_i} \) \( i = 1, 2, \ldots, n \)
\[ \text{Res}_i = \left| a_{ij} x_{i}^{\text{new}} + b_{ij} x_{j}^{\text{new}} + c_{ij} x_{k}^{\text{new}} - d_{ij} \right|, \ i = 1, 2, ..., n \]

* If \( \text{max} (\text{Res}_i) > \text{tol} \) then \( x^{\text{old}} = x^{\text{new}} \) & Repeat
  Else you are done!

2. Gauss-Seidel Method:

  ★ Same procedure as Jacobi Method except that

  \[ b_i^{*} = b_i - \sum d_{ij} x_j^{\text{new}} - \sum d_{ik} x_k^{\text{old}} \quad j < i \]
  \[ k > i \]

  * Gauss-Seidel can be twice faster than Jacobi but we lost parallel computation (\( x_j^{\text{new}} \) isn't calculated if \( x_j^{\text{old}} \) and \( x_j^{\text{new}} \) are all obtained)

3. Successive Over Relaxation: SOR

  1. Calculate \( \bar{x}_{65} = \frac{b_5}{a_{65}} \)

     Then \( x^{\text{new}} = x^{\text{old}} + \omega (\bar{x}_{65} - x^{\text{old}}) \)

     \[ 1 < \omega < 2 \]

     * Choosing \( \omega \) makes this method 10 times faster than Jacobi
Symmetric SOR:
\[ i = 1, 2, 3, \ldots, n \]

Periodic SOR:
(a) Update odd values
\[ i = 1, 3, 5, \ldots \]

(b) Update even values

Periodic Symmetric SOR:
# Nonlinear Problem:

\[ x \cdot x^2 - 4x = -3 \]

\[ \Rightarrow x_n \cdot x_n^2 - 4x_n = -3 \]

\[ \Rightarrow x_n = \frac{-3}{x_{n-1} - 4} \quad \text{&} \quad R_n = |x_n^2 - 4x_n + 3| \]

Iteration Process:

\[ n = 0 \quad \text{let} \quad x_0 = 0 \quad \Rightarrow \quad R_0 = 3 \]

\[ n = 1 \quad x_1 = \frac{-3}{-4} = 0.75 \quad \Rightarrow \quad R_1 = 0.5625 \]

\[ n = 2 \quad x_2 = \frac{-3}{0.75 - 4} = 0.923 \quad \Rightarrow \quad R_2 = 0.1599 \]

\[ n = 3 \quad x_3 = \frac{-3}{0.923 - 4} = 0.975 \quad \Rightarrow \quad R_3 = 0.0506 \]

\[ \ldots \]

As the iteration process continues, the residuary is getting smaller & smaller as \( x \to 1 \) which is one of the exact solution of this nonlinear equation.

NOTE: If you started with \( x_0 = 5 \), you would reached \( x = 3 \) which is the other solution!
# Solution Methods:

1. **Fixed Point Iteration:** "Converge linearly"

\[ f(x) = 0 \implies x = g(x^{n-1}) \]

*Convergence will occur if \( |g'(x)| < 1 \).*

2. **Newton's Method:** "Converge quadratically"

\[ x^n = x^{n-1} - \frac{f(x^{n-1})}{f'(x^{n-1})} \]

So if \( \delta = x^n - x^{n-1} \) then

\[ f' \delta = - \text{Res} \]

where \( \text{Res} = f(x^{n-1}) \)
3. Halley's Method = "Converge Cubic"

\[
x^n = x^{n-1} - \frac{f'(x^{n-1}) - \sqrt{f'(x^{n-1})^2 - 2 f(x^{n-1}) f''(x^{n-1})}}{f''(x^{n-1})}
\]

4. Chebychev Method = "between Newton's & Halley's"

\[
x^n = x^{n-1} \left[ \frac{f(x^{n-1})}{f'(x^{n-1})} - \frac{f(x^{n-1})^2}{2 f'(x^{n-1})^3} \right]
\]

*Note that Halley's Method won't be applied if \( f'(x^{n-1})^2 < 2 f(x^{n-1}) f''(x^{n-1}) \) or if \( f''(x^{n-1}) = 0 \)

While Newton's Method & Chebychev Method won't be applied if \( f'(x^{n-1}) = 0 \)!

# Systems of Nonlinear Equations: \( \mathbf{F}(\mathbf{x}) = 0 \)

1. Fixed Point Iteration:

\[
\begin{bmatrix}
\frac{\partial F_1}{\partial x_1} & 0 & \cdots & 0 \\
0 & \frac{\partial F_2}{\partial x_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{\partial F_n}{\partial x_n}
\end{bmatrix}
\begin{bmatrix}
\delta_1 \\
\delta_2 \\
\vdots \\
\delta_n
\end{bmatrix}
=
\begin{bmatrix}
R_1 \\
R_2 \\
\vdots \\
R_n
\end{bmatrix}
\]
2. Newton's Method:

\[ \bar{J} \bar{S} = - \bar{R} \]

Where

\[
J = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}
\end{bmatrix}
\]

Ex.: Solve: \( x^2 + y^2 = 1 \)
\[
\frac{x^2}{4} + \frac{y^2}{0.25} = 1
\]

Solutions:
\[
\begin{bmatrix}
x^n + y^n \\
x^{n-1} - y^{n-1}
\end{bmatrix} \begin{bmatrix}
x^n - x^{n-1} \\
y^n - y^{n-1}
\end{bmatrix} = \begin{bmatrix}
-\frac{(x^n)^2 - (y^n)^2}{4} \\
\frac{(x^n)^2 - (y^n)^2}{0.25}
\end{bmatrix}
\]

Initial Guess \( \Rightarrow \) Calculate \( \bar{R} \) \( \Rightarrow \) Calculate \( \bar{S} \)

\( \text{if} \ \max(\bar{R}) < \text{tol} \)

Problem Solved

*Note: Jacobian must be updated each iteration*
The Mean Value Theorem states:

\[ f'(t) = \frac{f(x) - f(x_0)}{x - x_0} \]

\( x_0 < t < x_1 \)

For fixed point iteration, we have:

\[ x_{n+1} = x_n + 2f(x_n) \]

\[ x_{n+1} = g(x_n) \quad x_n = g(x_{n-1}) \]

\[ x_{n+1} - x_n = g(x_n) - g(x_{n-1}) \]

\[ \frac{g(x_n) - g(x_{n-1})}{x_n - x_{n-1}} = \frac{x_{n+1} - x_n}{x_n - x_{n-1}} = g'(t) \]

So there is some \( t \) where \( x_n < t < x_{n+1} \) such that:

\[ g'(t) = \frac{S_{n+1}}{S_n} = \frac{g(x_n) - g(x_{n-1})}{x_n - x_{n-1}} \]

So if \( |g'(t)| < 1 \) \( \Rightarrow \) Convergence occurs.