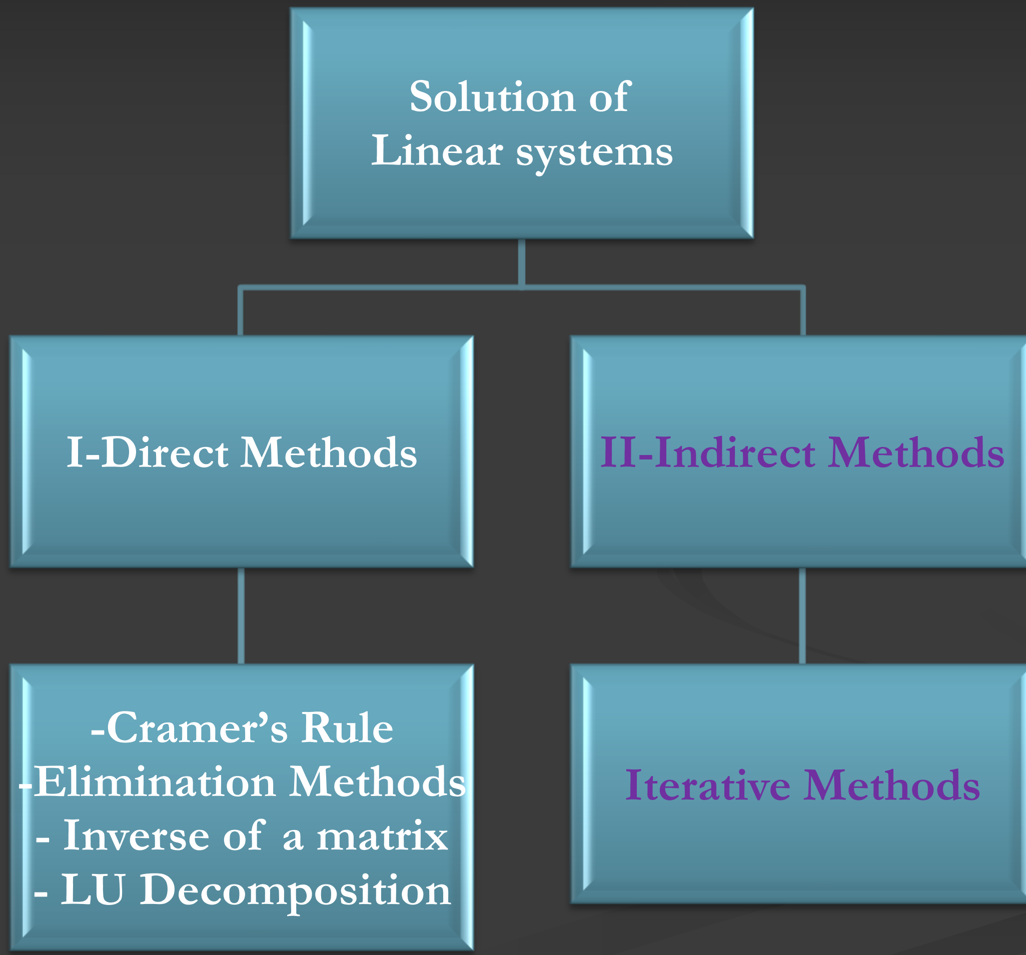
Solution of Systems of Linear Equations and Applications with MATLAB® :

II - Indirect Methods

Selis Önel, PhD

Solution methods for linear systems

$$A\,x = y$$



Iterative Solution

- Good for large systems of equations when Gauss elimination is NOT good,

  i.e., if n>>m for $|A_{m,n}||x_{n,1}| = |y_{m,1}|$

  (# unknowns is very large compared to # equations)
- Simple programming
- Applicable to nonlinear coefficients
- Requires an initial guess to start the iteration
- The goal is to:
  - Choose a good initial guess x0 for x
  - Substitute x0 in the equations and check if the right hand side of equations is equal to the left hand side or if $x-x0<\varepsilon$
  - Increment/decrement x0 until all equations are satisfied

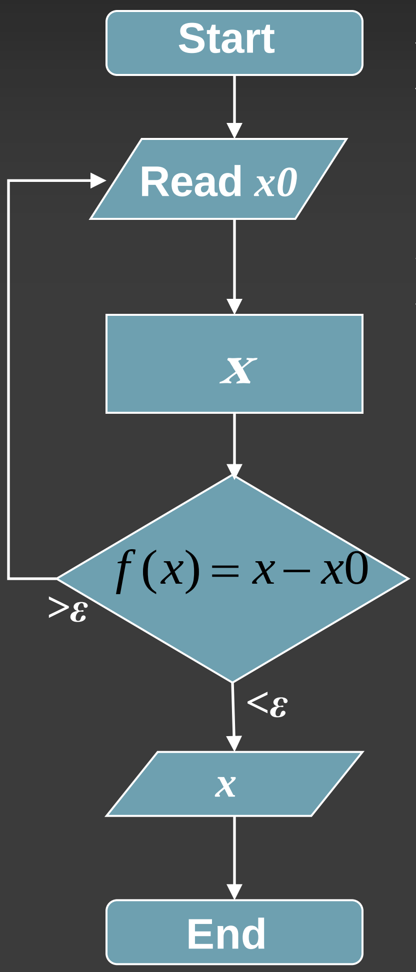
Iterative Solution

- Popular technique for finding roots of equations
- Applied to systems of linear equations to produce accurate results (Generalized *fixed point iteration*)
- Jacobi iteration: Carl Jacobi (1804-1851)
- Gauss-Seidel iteration: Johann Carl Friedrich Gauss (1777-1855) and Philipp Ludwig von Seidel (1821-1896)

Quotations

- It is true that Fourier had the opinion that the principal aim of mathematics was public utility and explanation of natural phenomena; but a philosopher like him should have known that the sole end of science is the honor of the human mind, and that under this title a question about numbers is worth as much as a question about the system of the world.
  Quoted in N Rose *Mathematical Maxims and Minims* (Raleigh N C 1988). Carl Jacobi
- There are problems to whose solution I would attach an infinitely greater importance than to those of mathematics, for example touching ethics, or our relation to God, or concerning our destiny and our future; but their solution lies wholly beyond us and completely outside the province of science.
  Quoted in J R Newman, *The World of Mathematics* (New York 1956). Carl Friedrich Gauss

A x = y Solution by Iteration



Input an initial guess for iteration to get started

- Can be any arbitrary vector x0

$$x0 = \begin{pmatrix} 0 \\ 0 \\ .. \end{pmatrix}$$

Ex: null vector x0=zeros(m,1)

- Good initial guess →fast **convergence**
- Consecutive solution of similar problems: Use the solution of previous problem as the initial guess for the next
- Iteration does not always converge!

A x = y Solution by Iteration: Convergence

Sufficient condition for iteration to converge:

- **Matrix A** should be **diagonally dominant**,

for all i: $$\left|a_{i,i}\right| > \sum_{j=1,\, j\neq i}^{n} \left|a_{i,j}\right|$$ or $$\left|a_{i,i}\right| > \sum_{j=1}^{i-1} \left|a_{i,j}\right| + \sum_{j=i+1}^{n} \left|a_{i,j}\right|$$

i.e. diagonal elements are larger in absolute value than the sum of the absolute value of other coefficients

- If A is irreducible (no part of the equation can be solved independently of the rest) for all i

Is it diagonally dominant ?

$$\begin{bmatrix} -2 & 1 & 6 \\ 4 & 7 & 1 \\ 3 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 15 \\ -10 \\ 5 \end{bmatrix}$$

- The matrix is <u>NOT</u> diagonally dominant

$$\begin{bmatrix} 3 & -1 & 1 \\ 4 & 7 & 1 \\ -2 & 1 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ -10 \\ 15 \end{bmatrix}$$

- The matrix is diagonally dominant

A x = y Solution by Iteration: Convergence

The iterative solution described here converges ***unconditionally*** if

- for a ***nonsingular matrix***, applied after premultiplying the equation Ax=y by A^t.

$$\boxed{A^t A x = A^t y}$$

Ex: Diagonally Dominant Matrix

Set of equations given by:

(1) $10x_1 - 2x_2 + 5x_3 = 8$

(2) $x_1 + 7x_2 - 3x_3 = 10$

(3) $-4x_1 - 2x_2 - 8x_3 = -20$

is predominantly diagonal as:

$|10| > |-2| + |5|$

$|7| > |1| + |-3|$

$|-8| > |-4| + |-2|$

→

$$Ax = y$$

$$\begin{pmatrix} 10 & -2 & 5 \\ 1 & 7 & -3 \\ -4 & -2 & -8 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \\ -20 \end{pmatrix}$$

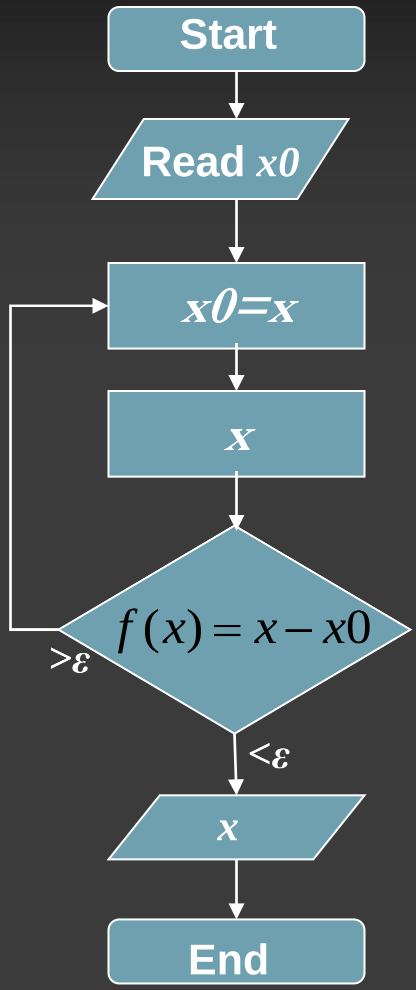
Unknown variables on the diagonal are given by:

$$x_1 = \frac{8-(-2x_2+5x_3)}{10}$$

$$x_2 = \frac{10-(x_1-3x_3)}{7}$$

$$x_3 = \frac{-20-(-4x_1-2x_2)}{-8}$$

A x = y Solution by Iteration: Convergence



- Initial guess values are used to calculate new guess values
- New estimates of x are calculated
- Iteration continues until convergence is satisfied, i.e. $f(x)<\varepsilon$

ε : convergence criteria (tolerance)

Jacobi (Simple) Iteration

(1) $a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = y_1$

(2) $a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = y_2$

..

(n) $a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = y_n$

$\sum_{j=1}^{n} a_{i,j}x_j = y_i$, where $i = 1, 2, \ldots, n$. Extracting x_i yields $a_{i,i}x_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j = y_i$

Solving for x_i gives: $$x_i = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j \right)$$

Consequently, the iterative scheme should be $$x_i \leftarrow \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j \right)$$

Jacobi (Simple) Iteration

Iteration cycle:

- Choose a starting vector x0 (Initial guesses)
- If a good guess for solution is not available, choose x randomly
- Use $x_i \leftarrow \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}x_j\right)$ with x_j =x0 to recompute each value of x

4. Check if |x-x0|<ε (tolerance), if so x=x0
5. If |x-x0|>ε, assign new values to x0

Repeat this cycle until changes in x (x-x0) between successive iteration cycles become sufficiently small, i.e, |x-x0|<ε

Jacobi (Simple) Iteration

$$x_i^{(t)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(t-1)} \right),$$ where t is the iteration count

for t=1

$$x_i^{(1)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(0)} \right),$$ where $x_j^{(0)}$ is the initial guess x0

if $\left| x_{\mathrm{i}}^{(1)} - x_i^{(0)} \right| > \varepsilon,$

$$x_i^{(2)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(1)} \right)$$

continue iteration until $\left| x_{\mathrm{i}}^{(t)} - x_i^{(t-1)} \right| \leq \varepsilon$ or $\left| \mathrm{y_i} - \left(a_{i,i} x_i^{(t)} - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} x_j^{(t)} \right) \right| \leq \delta$

Ex: Jacobi (Simple) Iteration

(1) $4x_1 - 2x_2 + x_3 = 3$

(2) $3x_1 - 7x_2 + 3x_3 = -2$

(3) $x_1 + 3x_2 - 5x_3 = -8$

$$\rightarrow \begin{pmatrix} 4 & -2 & 1 \\ 3 & -7 & 3 \\ 1 & 3 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -2 \\ -8 \end{pmatrix} \rightarrow ax = y$$

$$x_1 = \frac{3-(-2x_2+x_3)}{4}$$

$$x_2 = \frac{-2-(3x_1+3x_3)}{-7}$$

$$x_3 = \frac{-8-(x_1+3x_2)}{-5}$$

```
>> x0=zeros(n,1)
x0 =
     0
     0
     0
```

```
t =     1
x =   0.75000000000000
                     0
                     0
t =     1
x =   0.75000000000000
      0.28571428571429
                     0
t =     1
x =   0.75000000000000
      0.28571428571429
      1.60000000000000
```

```
t =     2
x =   0.49285714285714
      0.28571428571429
      1.60000000000000
t =     2
x =   0.49285714285714
      1.29285714285714
      1.60000000000000
t =     2
x =   0.49285714285714
      1.29285714285714
      1.92142857142857
```

Ex: Jacobi (Simple) Iteration

```
%Solve 3 strictly diagonally dominant linear equations for 3 unknowns: Jacobi iteration
a=[4 -2 1;3 -7 3;1 3 -5];   %Coefficient matrix
y=[3;-2;-8];                %Vector for values of f(x)=ax
n=length(y);
x=zeros(n,1);               %Create an empty matrix for x
x0=x;                       %Initial guess values for x
tmax=50;                    %Set max iteration no to stop iteration if system does not converge
tol=10^-3;                  %Set the tolerance to end iteration before t=tmax
for t=1:tmax,               %Start iteration
   for j=1:n,
         x(j)=(y(j)-a(j,[1:j-1,j+1:n])*x0([1:j-1,j+1:n]))/a(j,j);
   end
   error=abs(x-x0);   x0=x;
   if error<=tol
      ' Convergence is good. Iteration ended before tmax '
      break
   end
end
display('Iteration no=');  display(t-1);
x
```

Ex: Jacobi (Simple) Iteration

Results of the Jacobi iteration
in the command window:

```
ans =
Convergence is good. Iteration ended before tmax
Iteration no=
ans =
   18
x =
  1.00011187524906
  1.99949883459545
  2.99983186316654
```

Direct solution by Gauss elimination in the command window:

```
>> x=a\y
x =
     1
     2
     3
```

Gauss-Siedel Iteration

Iteration cycle:

- Choose a starting vector x0 (Initial guesses)
- If a good guess for solution is not available, choose x randomly
- Use $x_i^{(t)} \leftarrow \frac{1}{a_{i,i}}\left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(t)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(t-1)} \right)$ to compute each

element of x, always using the latest available values ox x_j

- Helps accelerate convergence
- Simplifies programming as the new values can be written over the old ones

Gauss-Siedel Iteration

$$x_i^{(t)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(t)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(t-1)} \right),$$ where t is the iteration count

for t=1

$$x_i^{(1)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(0)} \right),$$ where $x_j^{(0)}$ is the initial guess x0

and $x_j^{(1)}$ is the updated value calculated using $x_j^{(0)}$

if $\left| x_\mathrm{i}^{(1)} - x_i^{(0)} \right| > \varepsilon,$

$$x_i^{(2)} = \frac{1}{a_{i,i}}\left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(2)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(1)} \right)$$

continue iteration until $\left| x_\mathrm{i}^{(t)} - x_i^{(t-1)} \right| \le \varepsilon$ or $\left| \mathrm{y_i} - \left(a_{i,i} x_i^{(t)} - \sum_{j=1}^{i-1} a_{i,j} x_j^{(t)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(t-1)} \right) \right| \le \delta$

Gauss-Siedel Iteration with Relaxation: Successive Over Relaxation

To improve the convergence of Gauss-Siedel method using relaxation:

- Take the new value of x_i as a weighted average of its previous value and the predicted/calculated value

$$x_i^{(t)} = \omega \frac{1}{a_{i,i}} \left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(t)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(t-1)} \right) + (1-\omega) x_i^{(t-1)},$$

where

t : iteration count

ω : over-relaxation parameter satisfying $1 \le \omega < 2$

If ω=1, the SOR reduces to the Gauss-Siedel method

Successive Over-Relaxation: SOR

- If ω=1, no relaxation
- If ω<1, under-relaxation, i.e. interpolation between the old x_i and the calculated x_i
- If ω>1, over-relaxation, i.e. extrapolation
- A good estimate for an optimal value of ω can be computed during run time:

Let $\Delta x^{(k)} = \left| x^{(k-1)} - x^{(k)} \right|$ be the magnitude of the change in x during the kth iteration for ω=1 (without relaxation)

If k is sufficiently large, say k ≥ 5

$$\omega_{opt} \approx \frac{2}{1+\sqrt{1-\left(\dfrac{\Delta x^{(k+p)}}{\Delta x^{(k)}}\right)^{\frac{1}{p}}}},$$ where p is a positive integer

Ex: Gauss-Siedel with Relaxation (SOR)

```
%Solve 3 linear equations that are strictly diagonally dominant
%for 3 unknowns using  SOR iteration
a=[4 -2 1;3 -7 3;1 3 -5];      %Vector for values of f(x)=ax
y=[3;-2;-8];                    %Vector for values of f(x)=ax
n=length(y);
x=zeros(1,n);                   %Create an empty matrix for x
w=1.2;                          %Relaxation constant
for t=1:50,
    error=0;
    for i=1:n,
        s=0;    xb=x(i);
        for j=1:n,
            if i~=j,    s=s+a(i,j)*x(j);    end,
        end
        x(i)=w*(y(i)-s)/a(i,i)+(1-w)*x(i);
        error=error+abs(x(i)-xb);
    end
    fprintf('Iteration no = %3.0f, error = %7.2e \n', t, error)
    if error/n<10^-4,    break;    end
end,        x
```

Ex Cont^d.: Successive Over-Relaxation

```
Iteration no =   1, error = 4.42e+000
Iteration no =   2, error = 1.52e+000
Iteration no =   3, error = 1.12e+000
Iteration no =   4, error = 2.13e-001
Iteration no =   5, error = 9.29e-002
Iteration no =   6, error = 3.20e-002
Iteration no =   7, error = 1.21e-002
Iteration no =   8, error = 4.42e-003
Iteration no =   9, error = 1.63e-003
Iteration no =  10, error = 5.99e-004
Iteration no =  11, error = 2.20e-004

x =
   1.00004015934601   1.99999668943987   3.00001586803950
```