ECE 552
Numerical Circuit Analysis

Chapter Three

SOLUTION OF LINEAR ALGEBRAIC EQUATIONS

I. Hajj 2017

Linear Equation Solution Methods

Consider a set of **n** linear algebraic equations **Ax** = **b**

where **A** is a real or complex **n** x **n nonsingular** matrix.

There are many ways of solving **Ax** =**b**

I. **Direct Methods:** Cramer's Rule, Inverse, Gaussian Elimination or LU Factorization, Sparse Matrix Techniques

II. **Iterative or Relaxation Methods:** Gauss-Jacobi; Gauss-Seidel, Projection Methods ...

Dot or Inner Product

$$c = \mathbf{x}^{\mathrm{T}}\mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

Algorithm 3.1.1 Computation of Dot or Inner Product

1. $c = 0$
2. for $i = 1 : n$
3. $\quad c \Leftarrow c + x_i y_i$
4. end

Outer Product

$$\mathbf{C} = \mathbf{x}\mathbf{y}^{\mathrm{T}} = \begin{bmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_n \\ x_2y_1 & x_2y_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_my_1 & \vdots & & x_my_n \end{bmatrix}$$

The product $\mathbf{C} = \mathbf{x}\mathbf{y}^{\mathrm{T}}$ is also referred to as a *dyad*. It can be easily shown that $\mathbf{C}$ has rank 1.

Computation of Outer Product or Dyad $\mathbf{C} = \mathbf{x}\mathbf{y}^{\mathrm{T}}$

(***Row version***)

1. for $i = 1 : m$
2. for $j = 1 : n$
3. $\mathbf{C}(i, j) = \mathbf{x}(i)\mathbf{y}(j)$
4. end
5. end

which can also be written as

1. for $i = 1 : m$
2. $\mathbf{C}(i, :) = \mathbf{x}(i)\mathbf{y}(1 : n)$
3. end

Computation of Outer Product or Dyad $\mathbf{C} = \mathbf{x}\mathbf{y}^{\mathrm{T}}$

(*Column version*)

1. for $j = 1 : n$
2. for $i = 1 : m$
3. $\mathbf{C}(i, j) = \mathbf{x}(i)\mathbf{y}(j)$
4. end
5. end

which can also be written as

1. for $j = 1 : n$
2. $\mathbf{C}(:, j) = \mathbf{x}(1 : m)\mathbf{y}(j)$
3. end

(1) Cramer's Rule

$$x_k = \frac{\det [\mathbf{A} \text{ with Col. k replaced by } \mathbf{b}]}{\det [\mathbf{A}]}$$

Computationally very expensive.

(2) Compute the *inverse* $\mathbf{A^{-1}}$, then for every b, $\mathbf{x = A^{-1}b}$

Note: The inverse can be obtained by transforming:

$$[\mathbf{A} \vdots \mathbf{I}] \rightarrow [\mathbf{I} \vdots \mathbf{B}]$$

Then $\mathbf{A^{-1} \equiv B}$

Also computationally expensive, but much less than Cramer's Rule

Using the inverse: $\mathbf{x = A^{-1} b}$

- Computing $\mathbf{A^{-1}}$ requires n^3 operations (we'll prove this later)
- The computation of $\mathbf{x = A^{-1} b}$ requires n^2 operations.

Total: $n^3 + n^2$ operations

Storage: n^2 for **A,** n^2 for $\mathbf{A^{-1}}$, n for **b** and n for **x**; total = $2(n^2 + n)$

- An **operation** is considered a multiplication or a division, not counting additions and subtractions, which are almost as many as multiplications and divisions in solving **Ax = b.**

If **A** is sparse (i.e., the percentage of nonzero elements in **A** is small), then $\mathbf{A^{-1}}$, in general, is full and the solution process still requires $\mathbf{n^3 + n^2}$ operation and $\mathbf{2(n^2 + n)}$ storage locations.

A more efficient method, from both computation and storage, is to use

Gaussian Elimination or **LU factorization**

Gaussian Elimination

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

$a_{11} \neq 0$ $\Downarrow$

$$\left[\begin{array}{c|ccc} 1 & a_{12}/a_{11} & \cdots & a_{1n}/a_{11} \\ \hline 0 & a_{22} - \frac{a_{21}a_{12}}{a_{11}} & \cdots & a_{2n} - \frac{a_{21}a_{1n}}{a_{11}} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2} - \frac{a_{n1}a_{12}}{a_{11}} & \cdots & a_{nn} - \frac{a_{n1}a_{1n}}{a_{11}} \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \left[\begin{array}{c} b_1/a_{11} \\ \hline b_2 - \frac{a_{21}b_1}{a_{11}} \\ \vdots \\ b_n - \frac{a_{n1}b_1}{a_{11}} \end{array}\right]$$

Gaussian Elimination (Cont.)

$$\frac{1}{a_{22}^{(1)}} \times \left[\begin{array}{c|ccc} 1 & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ \hline 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array}\right] = \left[\begin{array}{c} b_1^{(1)} \\ \hline b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{array}\right]$$

$a_{22}^{(1)} \neq 0$

$$\begin{bmatrix} 1 & a^{(1)}{}_{12} & a^{(1)}{}_{12} & \cdots & a^{(1)}{}_{1n} \\ 0 & 1 & a^{(1)}{}_{23}/a^{(1)}{}_{22} & \cdots & a^{(1)}{}_{2n}/a^{(1)}{}_{22} \\ \vdots & 0 & \vdots & & \vdots \\ 0 & 0 & a^{(1)}{}_{n3} - a^{(1)}{}_{n2}\dfrac{a^{(1)}{}_{23}}{a^{(1)}{}_{22}} & \cdots & a^{(1)}{}_{nn} - \dfrac{a^{(1)}{}_{n2}a^{(1)}{}_{2n}}{a^{(1)}{}_{22}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1{}^{(1)} \\ b_2{}^{(1)}/a_{22}{}^{(1)} \\ \vdots \\ b_n{}^{(1)} - \dfrac{a_{n2}{}^{(1)}b_2{}^{(1)}}{a_{22}{}^{(1)}} \end{bmatrix}$$

Final result of elimination:

$$\begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & 1 & u_{23} & \cdots & u_{2n} \\ & 0 & 1 & \cdots & u_{3n} \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{bmatrix}$$

or **U x = z**

Solution of Ux = z by Backward Substitution:

$$\underbrace{\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & 1 & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & 1 & \ddots & \vdots \\ 0 & 0 & 0 & & 1 \end{bmatrix}}_{\text{U (upper triangular)}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix} \triangleq \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

$x_n = z_n$

$x_{n-1} = z_{n-1} - u_{n-1}, x_n$

$x_1 = z_1 - u_{12}x_2 - u_{13}x_3 \ldots -u_{1n}x_n$

Gaussian Elimination Algorithm

Append $\mathbf{b}$ to $\mathbf{A}$; that is $\mathbf{A}_p = [\mathbf{A}\vdots\mathbf{b}]$ or $\mathbf{A}_p(:, n+1) = \mathbf{b}$

1. for $k = 1 : n-1$
2. while $\mathbf{A}_p(k,k) \neq 0$
3. for $j = k+1 : n+1$
4. $\mathbf{A}_p(k,j) \Leftarrow \mathbf{A}_p(k,j)/\mathbf{A}_p(k,k)$
5. end
6. for $i = k+1 : n$
7. for $j = k+1 : n+1$
8. $\mathbf{A}_p(i,j) \Leftarrow \mathbf{A}_p(i,j) - \mathbf{A}_p(i,k)\mathbf{A}_p(k,j)$
9. end
10. end
11. end
12. $\mathbf{A}_p(n, n+1) \Leftarrow \mathbf{A}_p(n, n+1)/\mathbf{A}_p(n,n)$

Gaussian Elimination Algorithm

1. for $k = 1 : n-1$
2. while $\mathbf{A}_p(k,k) \neq 0$
3. $\mathbf{A}_p(k, k+1 : n+1) \Leftarrow \mathbf{A}_p(k, k+1 : n+1)/\mathbf{A}_p(k,k)$
4. $\mathbf{A}_p(k+1 : n, k+1 : n+1) \Leftarrow \mathbf{A}_p(k+1 : n, k+1 : n+1)$
$-\mathbf{A}_p(k+1 : n, k)\mathbf{A}_p(k, k+1 : n+1)$
5. end
6. $\mathbf{A}_p(n, n+1) \Leftarrow \mathbf{A}_p(n, n+1)/\mathbf{A}_p(n,n)$

$\mathbf{A}_p(k+1 : n, k)\mathbf{A}_p(k, k+1 : n+1)$ is a dyad

Backward Substitution for Solving $\mathbf{Ux} = \mathbf{z}$

1. $\mathbf{x}(n) = \mathbf{z}(n)$
2. for $i = n-1:1$
3. $\quad \mathbf{x}(i) = \mathbf{z}(i) - \mathbf{U}(i, i+1:n)\mathbf{z}(i+1:n)$
4. end

$\mathbf{U}(i, i+1 : n)\mathbf{z}(i+1 : n)$ is an inner product

1. for $i = n-1:1$
2. $\quad \mathbf{A}_p(i, n+1) \Leftarrow \mathbf{A}_p(i, n+1) - \mathbf{A}_p(i, i+1:n)\mathbf{A}_p(i+1:n, n+1)$
3. end

Suppose **b** changes, while **A** remains unchanged.

Question: Is it possible to solve for the new **x** without repeating ***all*** the elimination steps since the **U** matrix remains the same, and only **b**, and subsequently **z**, changes?

The answer is **YES**: Save the operations that operate on **b** to generate the new **z**.

How?

Save the ***lower*** part of the "triangular" matrix.

More formally

$$\underbrace{\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}}_{\mathbf{A}^{(1)}} = \underbrace{\begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & & \\ \vdots & & \ddots & \\ -\frac{a_{n1}}{a_{11}} & & & 1 \end{bmatrix}}_{\mathbf{L}_1^{-1}} \overbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}^{\mathbf{A}}$$

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} a_{11} & & & \\ a_{21} & 1 & & \\ \vdots & & \ddots & \\ a_{n1} & & & 1 \end{bmatrix}}_{\mathbf{L}_1} \underbrace{\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}}_{\mathbf{A}(1)}$$

$$\underbrace{\begin{bmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & 1 & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & a_{33}^{(2)} & \cdots & \\ & & \vdots & & \\ 0 & 0 & a_{n3}^{(2)} & & a_{nn}^{(2)} \end{bmatrix}}_{\mathbf{A}^{(2)}} = \underbrace{\begin{bmatrix} 1 & & & & \\ & \frac{1}{a_{22}^{(1)}} & & & \\ & \frac{-a_{22}^{(1)}}{a_{22}^{(1)}} & 1 & & \\ & \frac{-a_{n2}^{(1)}}{a_{22}^{(1)}} & & & 1 \end{bmatrix}}_{\mathbf{L}_2^{-1}} \underbrace{\begin{bmatrix} 1 & a_{12}^{(1)} & \cdots & \\ 0 & a_{22}^{(1)} & \cdots & \\ \vdots & \vdots & \ddots & \\ 0 & a_{n2}^{(1)} & & a_{nn}^{(1)} \end{bmatrix}}_{\mathbf{A}^{(1)} = \mathbf{L}_1^{-1}\mathbf{A}}$$

$$\mathbf{L}_2 = \begin{bmatrix} 1 & 0 & \cdots & \\ 0 & a_{22}^{(1)} & & 0 \\ & \vdots & 1 & \\ 0 & a_{n2}^{(1)} & & 1 \end{bmatrix} \Rightarrow \mathbf{A} = \mathbf{L}_1\mathbf{L}_2\mathbf{A}^{(2)}$$

$$\mathbf{A} = \underbrace{\mathbf{L}_1\mathbf{L}_2\cdots\mathbf{L}_{n-1}\mathbf{L}_n}_{\mathbf{L}} \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1n} \\ & 1 & u_{23} & \cdots & u_{2n} \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} a_{11} & & & 0 \\ a & a_{22}^{(1)} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2}^{(1)} & & a_{nn}^{(n-1)} \end{bmatrix}$$

$$\mathbf{A} = \mathbf{LU} = \underbrace{\begin{bmatrix} a_{11} & & & 0 \\ a_{21} & a_{22}^{(1)} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2}^{(1)} & \cdots & a_{nn}^{(n-1)} \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{bmatrix}}_{\mathbf{U}}$$

$$\mathbf{L} \equiv \begin{bmatrix} \ell_{11} & & 0 \\ \ell_{21} & \ell_{22} & \\ \vdots & \vdots & \ddots \\ \ell_{n1} & \cdots & \ell_{nn} \end{bmatrix}$$

$$\mathbf{A} \rightarrow \begin{bmatrix} \ell_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ \ell_{21} & \ell_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & \vdots & & \ddots & u_{n-1,\,n} \\ \ell_{n1} & \ell_{n2} & \cdots & \cdots & \ell_{nn} \end{bmatrix}$$

Both ***L*** and ***U*** are stored in place of ***A***

Summary of Solution Steps $\mathbf{Ax = b}$

- Factorize **A** = **LU** ➔ (**LUx** = **b**)
- Forward Substitution: **Lz = b**, where **Ux ≡ z**
- Backward Substitution: Solve **Ux = z**
- Factorization is always possible when **A** is nonsingular, provided pivoting is allowed. Pivoting will be explained later.
- Factorization is not unique
- There are different ways and different implementations, depending on computer architecture and on memory access. The solution, however, is unique.

Factorizing A into LU
Where U Has 1s on the Diagonal

1. for $k = 1 : n - 1$
2. while $\mathbf{A}(k, k) \neq 0$
3. $\mathbf{A}(k, k+1 : n) \Leftarrow \mathbf{A}(k, k+1 : n)/\mathbf{A}(k, k)$
4. $\mathbf{A}(k+1 : n, k+1 : n) \Leftarrow \mathbf{A}(k+1 : n, k+1 : n)$
$-\mathbf{A}(k+1 : n, k)\mathbf{A}(k, k+1 : n)$
5. end

Factorizing A into LU
Where L Has 1s on the Diagonal

1. for $k = 1 : n - 1$
2. Wwhile $\mathbf{A}(k, k) \neq 0$
3. $\mathbf{A}(k+1 : n, k) \Leftarrow \mathbf{A}(k+1 : n, k)/\mathbf{A}(k, k)$
4. $\mathbf{A}(k+1 : n, k+1 : n) \Leftarrow \mathbf{A}(k+1 : n, k+1 : n)$
$-\mathbf{A}(k+1 : n, k)\mathbf{A}(k, k+1 : n)$
5. end

Forward Substitution for Solving $\mathbf{Lz} = \mathbf{b}$ Where U Has *1*s on the Diagonal

(***Row version***)

1. $\mathbf{b}(1) \Leftarrow \mathbf{b}(1)/\mathbf{L}(1,1)$
2. for $i = 2 : n$
3. $\quad \mathbf{b}(i) \Leftarrow (\mathbf{b}(i) - \mathbf{L}(i, 1 : i-1)\mathbf{b}(1 : i-1))/\mathbf{L}(i,i)$
4. end

(***Column version***)

1. for $i = 1 : n-1$
2. $\quad \mathbf{b}(i) \Leftarrow \mathbf{b}(i)/\mathbf{L}(i,i)$
3. $\quad \mathbf{b}(i+1 : n) \Leftarrow \mathbf{b}(i+1 : n) - \mathbf{L}(i+1 : n, i)\mathbf{b}(i)$
4. end

Forward Substitution for Solving $\mathrm{Lz} = \mathrm{b}$
Where $\mathbf{L}$ Has *1*s on the Diagonal

(***Row version***)

1. for $i = 2 : n$
2. $\quad \mathbf{b}(i) \Leftarrow \mathbf{b}(i) - \mathbf{L}(i, 1 : i-1)\mathbf{b}(1 : i-1)$
3. end

(***Column version***)

1. for $i = 1 : n-1$
2. $\quad \mathbf{b}(i+1 : n) \Leftarrow \mathbf{b}(i+1 : n) - \mathbf{L}(i+1 : n, i)\mathbf{b}(i)$
3. end

Backward Substitution for Solving $\mathbf{Ux} = \mathbf{z}$ Where U Has *1*s on the Diagonal

(***Row version***)

1. for $i = n - 1 : 1$
2. $\quad \mathbf{z}(i) \Leftarrow \mathbf{z}(i) - \mathbf{U}(i, i+1 : n)\mathbf{z}(i+1 : n)$
3. end

(***Column version***)

1. for $i = n : 2$
2. $\quad \mathbf{z}(1 : i-1) \Leftarrow \mathbf{z}(1 : i-1) - \mathbf{U}(1 : i-1, i)\mathbf{z}(i)$
3. end

Backward Substitution for Solving $\mathbf{Ux} = \mathbf{z}$ Where L Has *1*s on the Diagonal

(***Row version***)

1. $\mathbf{z}(n) \Leftarrow \mathbf{z}(n)/\mathbf{U}(n,n)$
2. for $i = n-1:1$
3. $\quad \mathbf{z}(i) \Leftarrow (\mathbf{z}(i) - \mathbf{U}(i, i+1:n)\mathbf{z}(i+1:n))/\mathbf{U}(i,i)$
4. end

(***Column version***)

1. for $i = n:2$
2. $\quad \mathbf{z}(i) \Leftarrow \mathbf{z}(i)/\mathbf{U}(i,i)$
3. $\quad \mathbf{z}(1:i-1) \Leftarrow \mathbf{z}(1:i-1) - \mathbf{U}(1:i-1, i)\mathbf{z}(i)$
4. end
5. $\mathbf{z}(1) \Leftarrow \mathbf{z}(1)/\mathbf{U}(1,1)$

Remarks

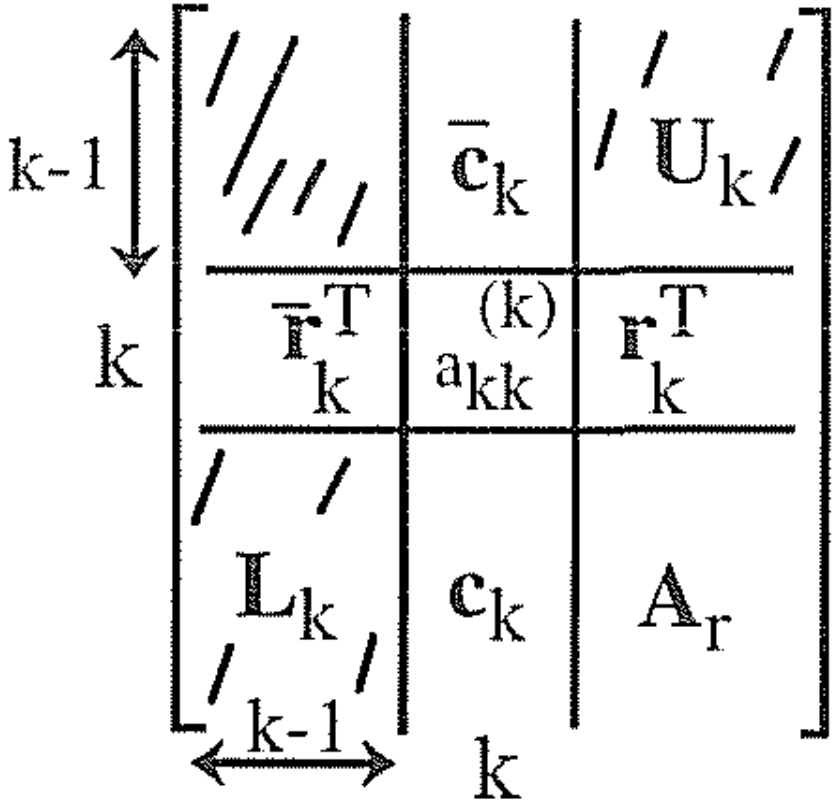
1. The matrices **L** and **U** can overwrite the values a_{ij} of **A**. There is no need to provide separate storage locations for **A** and for its factors **L** and **U**. There is no need to store the diagonal unit entries of **U** or **L**.

2. If only the right-hand-side vector **b** is changed, there is no need to repeat the factorization step; only the forward and backward substitutions need to be performed.

Determinant of **A**

- det **A** = det (**LU**) = det **L** det **U**
- det **A** = $l_{11} \times l_{22} \times \ldots \times l_{nn}$ (if **U** has 1s on diagonal
- det **A** = $u_{11} \times u_{22} \times \ldots \times u_{nn}$ (if **L** has 1s on diagonal

Other Factorization Procedures
(Doolittle's and Crout's)



Gauss Algorithm

at step k

$$\mathbf{r}_k^{\mathrm{T}} \Leftarrow \mathbf{r}_k^{\mathrm{T}} / a_{kk}^{(k)} \quad \text{or} \quad \mathbf{c}_k \Leftarrow \mathbf{c}_k / a_{kk}^{(k)}$$

$$\text{then} \qquad \mathbf{A}_r^{(k+1)} \Leftarrow \mathbf{A}_r^{(k)} - \mathbf{c}_k \mathbf{r}_k^{\mathrm{T}}$$

Doolittle Algorithm

$\mathbf{L}$ has 1s on diagonal

at step k

$$u_{kk} = a_{kk} - \bar{\mathbf{r}}_k^{\mathrm{T}} \bar{\mathbf{c}}_k$$

$$\mathbf{u}_k^T = \mathbf{r}_k^T - \bar{\mathbf{r}}_k^{\mathrm{T}} \mathbf{U}_k$$

$$\mathbf{l}_k = (\mathbf{c}_k - \mathbf{L}_k \bar{\mathbf{c}}_k)/u_{kk}$$

Crout Algorithm

$\mathbf{U}$ has ls on diagonal

at step k

$$\ell_{kk} = a_{kk} - \bar{\mathbf{r}}_k^{\mathrm{T}} \bar{\mathbf{c}}_k$$

$$\mathbf{l}_k = \mathbf{c}_k - \mathbf{L}_k \bar{\mathbf{c}}_k$$

$$\mathbf{u}_k^T = (\mathbf{r}_k^T - \bar{\mathbf{r}}_k^{\mathrm{T}} \mathbf{U}_k)/\ell_{kk}$$

Symmetric Systems of Equations

Cholesky Algorithm for Factorizing a Numerically Symmetric Positive Definite Matrix A into $\mathbf{LL}^{\mathrm{T}}$ Where A is Stored as a Lower Triangular Matrix

1. for $k = 1 : n - 1$
2. $\quad \mathbf{A}(k,k) = \sqrt{\mathbf{A}(k,k)}$ {positive square root}
3. $\quad \mathbf{A}(k+1:n,k) \Leftarrow \mathbf{A}(k+1:n,k)/\mathbf{A}(k,k)$
4. $\quad$ for $j = k+1 : n$
5. $\quad\quad \mathbf{A}(j:n,j) \Leftarrow \mathbf{A}(j:n,j) - \mathbf{A}(k,j)\mathbf{A}(j:n,k)$
6. $\quad$ end
7. end
8. $\mathbf{A}(n,n) = \sqrt{\mathbf{A}(n,n)}$

Solution of $\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$

{Forward substitution, column version}

1. for $i = 1 : n-1$
2. $\quad \mathbf{b}(i) \Leftarrow \mathbf{b}(i)/\mathbf{L}(i,i)$
3. $\quad \mathbf{b}(i+1:n) \Leftarrow \mathbf{b}(i+1:n) - \mathbf{L}(i+1:n,i)\mathbf{b}(i)$
4. end
5. $\mathbf{b}(n) \Leftarrow \mathbf{b}(n)/\mathbf{L}(n,n)$

Solution of $\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$

Backward substitution row version

1. $\mathbf{b}(n) \Leftarrow \mathbf{b}(n)/\mathbf{L}(n,n)$
2. for $i = n-1:1$
3. $\quad \mathbf{b}(i) \Leftarrow (\mathbf{b}(i) - \mathbf{L}(i+1:n,i)\mathbf{z}(i+1:n))/\mathbf{L}(i,i)$
4. end

Factorizing a Numerically Symmetric Matrix A into $\mathbf{LDL}^{\mathrm{T}}$, Where A is Stored as a Lower Triangular Matrix

1. for $k = 1 : n - 1$
2. $\quad \mathbf{v}(k+1:n) = \mathbf{A}(k+1:n, k)$
3. $\quad$ while $\mathbf{A}(k,k) \neq 0$
4. $\quad \mathbf{A}(k+1:n, k) \Leftarrow \mathbf{A}(k+1:n, k)/\mathbf{A}(k,k)$
5. $\quad$ for $j = k+1 : n$
6. $\quad\quad \mathbf{A}(j:n, j) \Leftarrow \mathbf{A}(j:n, j) - \mathbf{v}(j)\mathbf{A}(j:n, k)$
7. $\quad$ end
8. end

solution of $\mathbf{LDL}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$

1. Solve $\mathbf{Ly} = \mathbf{b}$ {forward substitution}
2. Solve $\mathbf{Dz} = \mathbf{y}$ or $z_i = y_i/d_{ii},\ i = 1, \ldots, n$
3. Solve $\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{z}$ {backward substitution}

Transpose Systems

- The solution of the transpose system $\mathbf{A}^T\mathbf{x} = \mathbf{c}$ can be found by using the same **LU** factors of **A.**
- $\mathbf{A}^T\mathbf{x} = (\mathbf{LU})^T\mathbf{x} = \mathbf{U}^T\mathbf{L}^T\mathbf{x} = \mathbf{c}$
- Forward substitution: Solve $\mathbf{U}^T\mathbf{z} = \mathbf{c}$
- Backward substitution: Solve $\mathbf{L}^T\mathbf{x} = \mathbf{z}$

Remark

- Transpose systems are used in small-change sensitivity calculations and optimization.

Operation Count

Useful identities

$$\sum_{j=0 \text{ or } 1}^{n} j = \frac{n(n+1)}{2}$$

$$\sum_{j=0 \text{ or } 1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$$

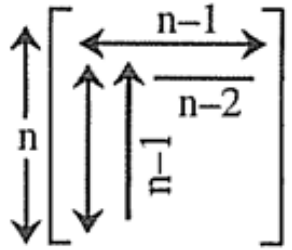
$$\sum_{j=0 \text{ or } 1}^{n-1} j = \frac{n(n-1)}{2}$$

$$\sum_{j=0 \text{ or } 1}^{n-1} j^2 = \frac{n(n-1)(2n-1)}{6}$$

Number of operations in LU factorization

➔ (assume L and U are full)

$$n(n-1)+(n-1)(n-2)+\ldots = \sum_{k=1}^{n}(n-k+1)(n-k)$$



Let $j = (n-k)$

$$= \sum_{j=0}^{n-1}(j+j^2) = \frac{n(n-1)}{2} + \frac{n(n-1)(2n-1)}{6}$$

$$= \frac{n(n-1)}{2}\left(1+\frac{2n-1}{3}\right) = \frac{n}{3}^{(n^2-1)} = \frac{n^3}{3} - \frac{n}{3}$$

Count divisions and multiplications only.
Number of additions and subtractions is almost equal to the number of divisions and multiplications

Forward and Backward Substitutions

$$\underbrace{\mathbf{L}\mathbf{z} = \mathbf{b} \quad \& \quad \mathbf{U}\mathbf{x} = \mathbf{z}}_{n^2 \text{ operations}}$$

$$\text{Total: } \frac{n^3}{3} - \frac{n}{3} + n^2$$

Pivoting

(1) For Numerical Accuracy and Stability

(2) For Sparsity Preservation

(1) Pivoting for numerical stability

(a) To avoid division by zero

Example

$$
\begin{aligned}
x_1 + x_2 + 2x_3 &= 1 \\
x_1 + x_2 + 3x_3 &= 0 \\
x_1 - x_2 + 4x_3 &= 0
\end{aligned}
$$

$$
\begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 3 \\ 1 & -1 & 4 \end{bmatrix} \rightarrow \left[\begin{array}{c|cc} 1 & 1 & 2 \\ \hline 1 & 0 & (3-2) \\ 1 & (-2) & 2 \end{array}\right]
$$

Pivoting for numerical stability (cont.)

- Pivoting for numerical stability to offset computation errors caused by round-off errors due to computer word length.

- WILKINSON: “To minimize round-off errors in Gaussian elimination it is important to avoid growth in the size of $a_{ij}(k+1)$ and $b_i(k+1)$.”
  That is, the pivot should not be ‘too’ small.

Different types of pivoting

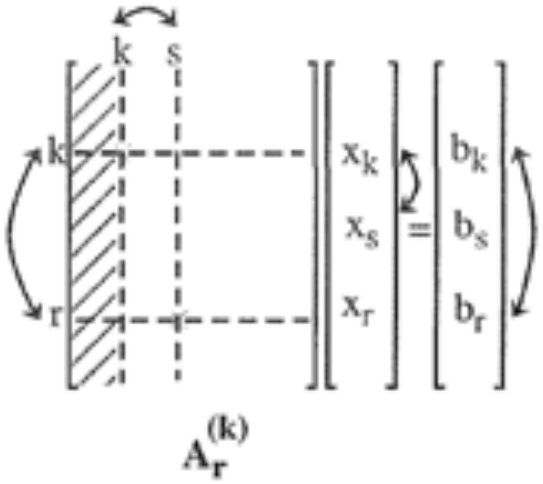
(1) Complete pivoting: Ax = b

At Step k in the factor process, choose $a_{rs}^{(k)}$ in the reduced matrix as a pivot such that

$$\left|a_{rs}^{(k)}\right| = \max\left|a_{ij}^{(k)}\right|, \quad i,j \geq k$$

This is done by row and column interchange to put $a_{rs}^{(k)}$ in position $a_{kk}^{(k)}$

(Pivoting can be accomplished by using a pointer system rather than actual row and column interchanges.)



- Complete pivoting can be used with GAUSS’ algorithm, but not with CROUT’S and DOOLITTLE’S.

- In complete pivoting the order of the variables and the rhs may change according to the reordering.

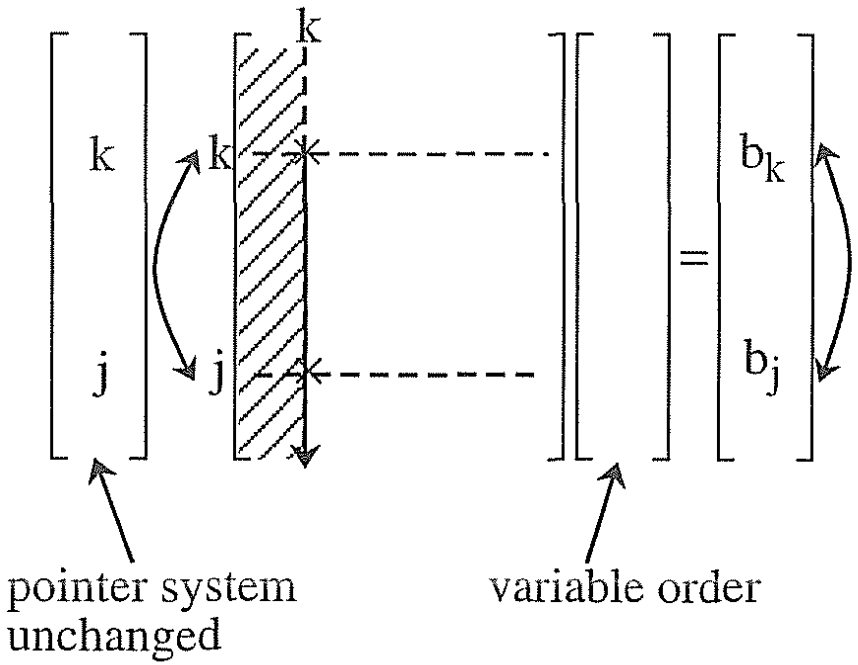
Partial Pivoting

(2) Row Pivoting

At the k-th step in the factorization process choose

$$\left|a_{kk}^{(k)}\right| \leftarrow \max \left|a_{jk}^{(k)}\right| \quad j \geq k.$$

i.e., choose the largest element in **column k** as a pivot by performing **row interchange**.



Different types of pivoting (cont.)

- In row pivoting the order of the variables remains the same, while the order of the rhs changes accordingly.

- Row pivoting can be used with GAUSS', CROUT'S, and DOOLITTLES.

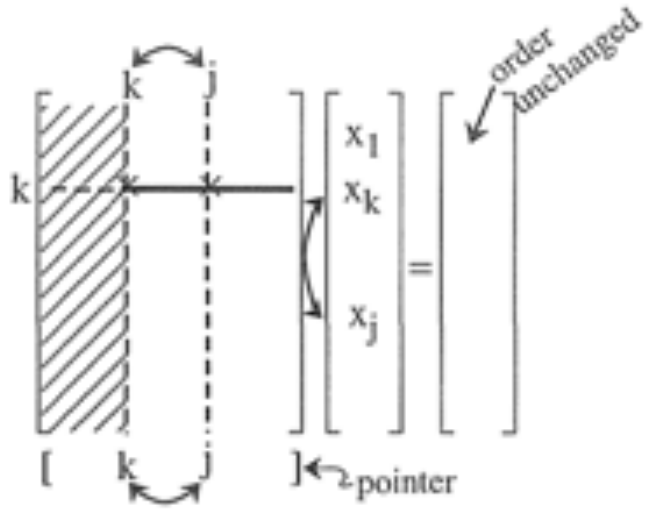
Different types of pivoting (cont.)

(3) Column Pivoting

At the k-th step in the factorization process choose as a pivot:

$$\left|a_{kk}^{(k)}\right| \leftarrow \max \left|a_{jk}^{(k)}\right| \quad j \geq k$$

i.e., choose the largest element in row k.
This is done by performing column interchange.



Different types of pivoting (cont.)

- In column pivoting the order of the variables changes, while the order of the rhs remains the same.

- Column pivoting can be used with GAUSS', CROUT'S, and DOOLITTLE'S.

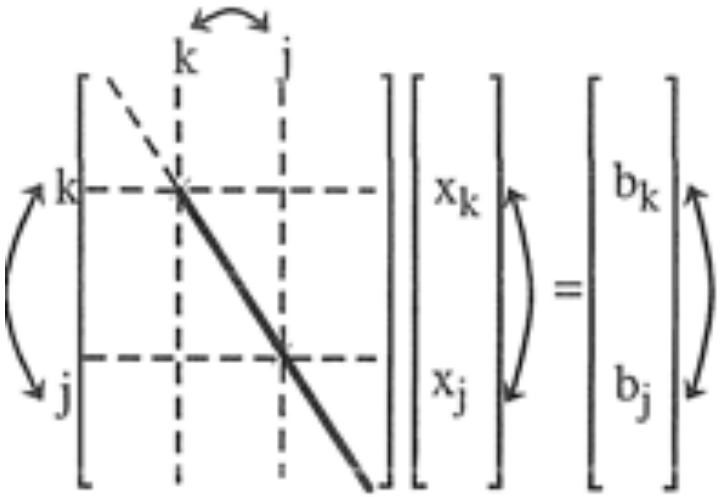
Different types of pivoting (cont.)

(4) Diagonal Pivoting

At the k-th step in the factorization process, choose

$$\left|a_{kk}^{(k)}\right| \leftarrow \max \left|a_{jj}^{(k)}\right| \quad j \geq k$$

i.e., choose the largest element along the diagonal of the reduced matrix.



Different types of pivoting (cont.)

- **Diagonal pivoting** requires both row and column interchanges in a symmetrical way. It is used, for example, to preserve symmetry. Both the orders of the variables and the rhs change accordingly.
- **Diagonal pivoting** can be used with GAUSS', but not with CROUT'S or DOOLITTLE'S.
- **Diagonally dominant matrices** require no pivoting to maintain numerical stability. The reduced matrix remains diagonally dominant.
- Pivoting, however, may be necessary to maintain **sparsity** (explained later).

Different types of pivoting (cont.)

•The nodal admittance matrix of a circuit with no controlled sources is diagonally dominant (e.g., power system load flow equations).

•**Diagonal dominance** depends on equation and variable ordering, i.e., diagonal dominance may be destroyed (or created) by pivoting (other than diagonal pivoting).
Example:

$$\begin{bmatrix} 3 & 1 \\ 4 & 5 \end{bmatrix} \xleftrightarrow{\text{row pivoting}} \begin{bmatrix} 4 & 5 \\ 3 & 1 \end{bmatrix}$$

diag. dom. not diag. dom.

Why is pivoting possible?

- **Theorem:** If at any step in the factorization process, a column or a row of the reduced matrix is zero, then **A** is singular.

- In other words, if **A** is nonsingular, then it is always possible to carry out complete, row, or column pivoting (but not necessarily diagonal pivoting).

- In some cases all the nonzero elements in the reduced matrix are small => matrix could be ill-conditioned.

- **Advice**: Always use double-precision

Pivoting and Permutation Matrix

- Row pivoting amounts to multiplying matrix **A** by a permutation matrix **P:**

$$\mathbf{PAx} = \mathbf{Pb}$$

- **P** is a reordered identity matrix
- Example

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

P can be encoded as a row vector $\mathbf{p}_r$ =[2 4 1 3]

Pivoting and Permutation Matrix

- Column pivoting amounts to post-multiplying matrix **A** by a permutation matrix **Q**:

AQ

- **Q** is a reordered identity matrix
- Example

$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Q can be encoded as a row vector $\mathbf{q_c}$ =[3 1 4 2]

Pivoting and Permutation Matrix

- Complete pivoting

$$\mathbf{PAQ}$$

- Diagonal pivoting

$$\mathbf{PAP}^T$$

Condition Number

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

If $\kappa(\mathbf{A})$ is small, **A** is well-conditioned

=> Solution can be obtained accurately

If $\kappa(\mathbf{A})$ Is large, **A** is ill-conditioned

=> Solution not accurate

Residual Error and Accuracy

- Residual error: $\mathbf{r} = \mathbf{b} - A\mathbf{x}^{(s)}$

  $\Rightarrow$Small $||\mathbf{r}||$ indicates convergence
- Accuracy is related to number of correct decimal digits in solution, related to condition number

- In circuit simulation small $||\mathbf{r}||$ is more important, and Gaussian elimination with pivoting is adequate.

Improving Accuracy

Algorithm 4.3.1: *Iterative Improvement Algorithm*

Suppose $\hat{\mathbf{x}}$ is the computed solution of $\mathbf{Ax} = \mathbf{b}$ that has been found by applying LU factorization with partial pivoting, $\mathbf{PA} = \mathbf{LU}$.

1. Find $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$
2. Solve $\mathbf{Ly} = \mathbf{Pr}$
3. Solve $\mathbf{Uz} = \mathbf{y}$
4. $\mathbf{x}_{\text{new}} = \hat{\mathbf{x}} + \mathbf{z}$
5. Repeat until the number of desired number of digits in $\mathbf{x}_{\text{new}}$ converge.

Solution Updating

(used in large-change sensitivity calculations and other applications)

- Suppose the **LU** factors of **A** and the solution $\mathbf{x}_o$ of **Ax = b** are known.

- Suppose **A** is "perturbed" (or some entries changed); find the ***new*** solution from **LU** and $\mathbf{x}_o$.

$$\text{Let } \mathbf{A}_{new} = \mathbf{A} + \mathbf{PDQ}^T \quad \text{nonsingular } \mathbf{D}$$

$$\mathbf{A} \text{ is } n \times n, \mathbf{P} \text{ and } \mathbf{Q} \text{ are } n \times k, \mathbf{D} \text{ is } k \times k, \mathbf{PDQ}^T \text{ has rank } k.$$

$$\text{Assume } k << n.$$

Sherman-Morrison-Woodbury (Householder) **Formula**

$$\mathbf{A}_{\text{new}}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}(\mathbf{D}^{-1} + \mathbf{Q}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{P})^{-1}\,\mathbf{Q}^{\mathrm{T}}\mathbf{A}^{-1}$$

(We'll prove later)

New Solution of $\mathbf{A}_{\text{new}}\,\mathbf{x} = \mathbf{b}$:

$$\mathbf{x} = (\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}(\mathbf{D}^{-1} + \mathbf{Q}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{P})^{-1}\,\mathbf{Q}^{\mathrm{T}}\mathbf{A}^{-1})\mathbf{b}$$

$$= \underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\mathbf{x}^{\circ}} - \mathbf{A}^{-1}\mathbf{P}(\mathbf{D}^{-1} + \mathbf{Q}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{P})^{-1}\mathbf{Q}^{\mathrm{T}}\underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\mathbf{x}^{\circ}}$$

$$\mathbf{x}_{\text{new}} = \mathbf{x}^{\circ} - (\mathbf{A}^{-1}\mathbf{P})\left(\mathbf{D}^{-1} + \mathbf{Q}^{\mathrm{T}}(\mathbf{A}^{-1}\mathbf{P})\right)^{-1}\mathbf{Q}^{\mathrm{T}}\mathbf{x}^{\circ}$$

Solution Updating Algorithm

- Given $\mathbf{x}^{\circ}$ and **LU** factors of **A**
- (1) Solve $\mathbf{AV = P \Rightarrow V = A^{-1}P}$
- (2) Form $\mathbf{(D^{-1} + Q^{T}V) \equiv H}$
- (3) Form $\mathbf{y = Q^{T}x^{o}}$
- (4) Solve $\mathbf{Hz = y}$ $\leftarrow$ k x k system
- (5) $\mathbf{x_{new} = x^{o} - Vz}$

Proof of Housholder formula

$$\left[\begin{array}{c|c} A & P \\ \hline Q^T & -D^{-1} \end{array}\right]\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

$\Downarrow$

$$\begin{bmatrix} A + PDQ^T & 0 \\ Q^T & -D^{-1} \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

$$\left[\begin{array}{c|c} LU & A^{-1}P \\ \hline 0 & -D^{-1} - Q^T A^{-1} P \end{array}\right]\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^{-1}b \\ -Q^T A^{-1} b \end{bmatrix}$$

OR

$$\begin{aligned} (-D^{-1} - Q^T A^{-1} P)z &= -Q^T A^{-1} b \\ &= -Q^T x^{\circ} \end{aligned}$$

$$\underbrace{(D^{-1} + Q^T A^{-1} P)z}_{H} = +y \quad \leftarrow$$

Back substitute:

$$x = x^{\circ} - \underbrace{(A^{-1}P)z}_{V}$$

Sensitivity Analysis of Linear Resistive Circuits and Linear RLC Circuits in the Frequency Domain: Linear Algebraic Systems

Applications:

- Design,
- optimization,
- tolerance,
- statistical analysis,
- reliability studies,
- failure analysis,
- .....

Large-Change Sensitivity

- Let F(**p**) represent a scalar output function of a system, where **p** is the set of system parameters. Let $\mathbf{p}^\circ$ be the nominal value of **p**.

- $\mathbf{F(p^\circ + \Delta p) - F(p^\circ)}$ **Large-change sensitivity** of F(**p**) with respect to **p** at $\mathbf{p} = \mathbf{p}^\circ$

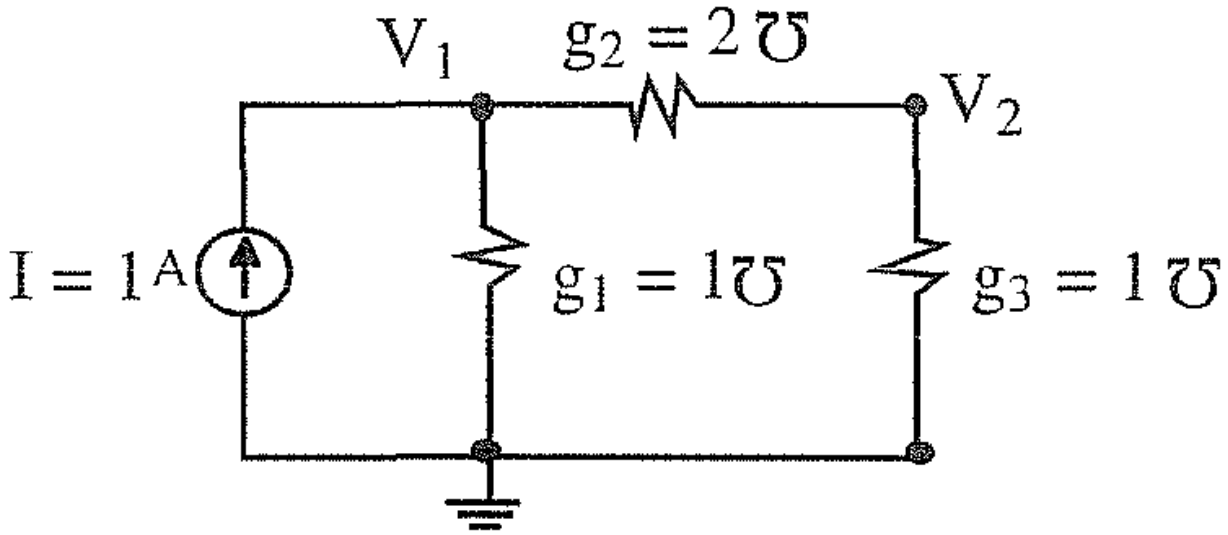
- **Apply solution updating method**

Solution Updating Algorithm

- Given $\mathbf{x}^\circ$ and **LU** factors of **A**
- (1) Solve $\mathbf{AV = P \Rightarrow V = A^{-1}P}$
- (2) Form $\mathbf{(D^{-1} + Q^T V) \equiv H}$
- (3) Form $\mathbf{y = Q^T x^\circ}$
- (4) Solve $\mathbf{Hz = y}$ ← k x k system
- (5) $\mathbf{x_{new} = x^\circ - Vz}$, **or**

$$\mathbf{x_{new} - x^\circ = -Vz \text{ (change in the solution)}}$$

Example on Large-Change Sensitivity Calculation



- Find large-change sensitivity of V_1 and V_2 wrt Δg_2.

Find the "Nominal Solution":

Example on Large-Change Sensitivity Calculation (cont.)

$$\underbrace{\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}}_{\mathbf{Y}} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 3 & 0 \\ -2 & 5/3 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 1 & -2/3 \\ 0 & 1 \end{bmatrix}}_{\mathbf{U}} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} V_1^0 \\ V_2^0 \end{bmatrix} = \begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix} \leftarrow \text{nominal solution}$$

$$\mathbf{Y}' = \mathbf{Y} + \begin{bmatrix} +\Delta g_2 & -\Delta g_2 \\ -\Delta g_2 & +\Delta g_2 \end{bmatrix}$$

$$= \mathbf{Y} + \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{\mathbf{P}} \Delta g_2 \underbrace{\begin{bmatrix} 1 & -1 \end{bmatrix}}_{\mathbf{Q}^T}$$

$$\begin{bmatrix} \mathbf{Y} & \mathbf{P} \\ \mathbf{Q}^T & -\Delta g_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{new} \\ \hline \mathbf{z} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \hline 0 \end{bmatrix} \} \mathbf{b}$$

Example on Large-Change Sensitivity Calculation (cont.)

$$\begin{bmatrix} & \overbrace{\quad\quad}^{\tilde{\mathbf{P}}} \\ \mathbf{I} & (\mathbf{Y}^{-1}\mathbf{P}) \\ 0 & (-\Delta g_2^{-1} - \mathbf{Q}^{\mathrm{T}}\underbrace{\mathbf{Y}^{-1}\mathbf{P}}_{\tilde{\mathbf{P}}}) \end{bmatrix} \begin{bmatrix} \mathbf{V}^{\mathrm{new}} \\ \mathbf{z} \end{bmatrix} = \underbrace{\begin{bmatrix} (\mathbf{Y}^{-1}\mathbf{b}) \\ -\mathbf{Q}^{\mathrm{T}}\mathbf{V}^{0} \end{bmatrix}}$$

$$(1) \qquad \underbrace{[\mathbf{Y}]}_{\mathbf{LU}}\tilde{\mathbf{P}} = \mathbf{P} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 3 & 0 \\ -2 & 5/3 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} 1 & -2/3 \\ 0 & 1 \end{bmatrix}}_{\mathbf{U}} \tilde{\mathbf{P}} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\tilde{\mathbf{P}} = \begin{bmatrix} -13/15 \\ -9/5 \end{bmatrix}$$

Example on Large-Change Sensitivity Calculation (cont.)

(2)

$$\mathbf{Q}^{\mathrm{T}}(\underbrace{\mathbf{Y}^{-1}\mathbf{P}}_{\tilde{\mathbf{P}}}) = \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} -13/15 \\ -9/5 \end{bmatrix}$$

$$= (14/15)$$

$$(\mathbf{Q}^{\mathrm{T}}\mathbf{V}^{0}) = \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix} = \left(\frac{1}{5}\right)$$

(3) $$-\left(\Delta g_2^{-1} + \frac{14}{15}\right)\mathbf{z} = \frac{-1}{5}\Bigg\}\text{Reduced system}$$

If $\Delta g2 = 1$,

$$\left(1 + \frac{14}{15}\right)\mathbf{z} = \frac{1}{5}$$

$$\mathbf{z} = 3/29$$

(4) $$\mathbf{V}^{\text{new}} = \underbrace{\mathbf{V}^{0}}_{\begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix}} - \underbrace{\tilde{\mathbf{P}}}_{\begin{bmatrix} -13/15 \\ -9/5 \end{bmatrix}}\mathbf{z} = \begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix} - \begin{bmatrix} -13/15 \\ -9/5 \end{bmatrix} \times \frac{3}{29}$$

Small-Change Sensitivity

$$\left.\frac{\partial F(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}^\circ} \triangleq \lim_{\Delta\mathbf{p}\to 0} \frac{F(\mathbf{p}^\circ+\Delta\mathbf{p})-F(\mathbf{p})}{\Delta\mathbf{p}}$$

- small-change sensitivity of F(**p**) with respect to **p** at **p** = **p**°

  If F(**p**) is available as an **explicit** function of **p**, then the sensitivities, both small and large, could be computed in a straight-forward manner. Usually, such an explicit function is *not available.*

- Consider a linear algebraic system described by:

  A(p)x = b

  where **A** could be real or complex.

Small-Change Sensitivity (Cont.)

The matrix $\left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right]$ is defined as the small-change sensitivity matrix.

- A column $\frac{\partial \mathbf{x}}{\partial p_j}$ is the sensitivity of all the variables **x** with respect to one parameter p_j.

A row $\frac{\partial x_i}{\partial \mathbf{p}}$ is the sensitivity of a single variable with respect to all parameters.

Suppose the output is a linear combination of the system variables:

$$x_{out} = \mathbf{e}^T \mathbf{x}$$

$$\text{Then } \frac{\partial x_{out}}{\partial \mathbf{p}} = \mathbf{e}^T \left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right]$$

a linear combination of the rows of the sensitivity matrix. The output could also be a vector $\mathbf{x}_{out} = \mathbf{E}^T\mathbf{x}$.

Small-Change Sensitivity (Cont.)

Differentiate system equations **A(p) x = b** with respect to **p:**

$$\left[\frac{\partial \mathbf{A}}{\partial \mathbf{p}}\right]\mathbf{x}+\mathbf{A}\left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right]=\frac{\partial \mathbf{b}}{\partial \mathbf{p}}$$

or

$$\mathbf{A}\left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right]=\frac{\partial \mathbf{b}}{\partial \mathbf{p}}-\left[\frac{\partial \mathbf{A}}{\partial \mathbf{p}}\right]\overset{\leftrightarrow}{\mathbf{x}}{}^{\circ}$$

(evaluated at nominal solution **x = x°** , **p = p°**)

Suppose $x_{out} = \mathbf{e}^T\mathbf{x}$, then

$$\frac{\partial x_{out}}{\partial \mathbf{p}} = \mathbf{e}^T\left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right]$$

$$= \underbrace{\mathbf{e}^T\mathbf{A}^{-1}}_{\mathbf{U}^T}\left[\frac{\partial \mathbf{b}}{\partial \mathbf{p}}-\frac{\partial \mathbf{A}}{\partial \mathbf{p}}\mathbf{x}^{\circ}\right]$$

Algorithm

1. Solve **Ax = b** at **p = p°** and find LU and **x°**

2. Construct

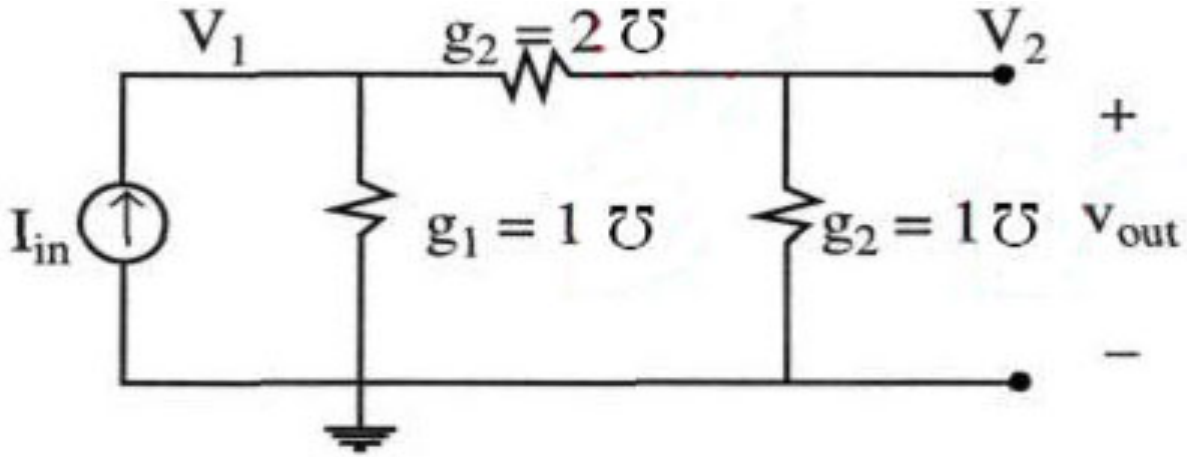
$$\mathbf{r}\Big|^{\overset{k}{\leftrightarrow}}_{\mathbf{B}} = \frac{\partial \mathbf{b}}{\partial \mathbf{p}} - \left[\frac{\partial \mathbf{A}}{\partial p_1}\mathbf{x}^\circ \cdots \frac{\partial \mathbf{A}}{\partial p_k}\mathbf{x}^\circ\right]$$

3. Solve the adjoint system (or transpose)

   $\mathbf{A}^T\mathbf{u} = \mathbf{e} \rightarrow \mathbf{U}^T\mathbf{L}^T\mathbf{u} = \mathbf{e}$ **(i.e., $\mathbf{u}^T = \mathbf{e}^T\mathbf{A}^{-1}$)**

4. Find $\dfrac{\partial x_{out}}{\partial \mathbf{p}} = \mathbf{u}^T\mathbf{B}$

Example



- Find the small-change sensitivity of v_{out} with respect to g_1, g_2, and g_3.

- Suppose we use nodal analysis, then

$$v_{out} = \underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{e^T} \underbrace{\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}}_{\mathbf{x}}$$

Example (cont.)

(1) Find nominal solution:

$$\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \end{bmatrix}\Rightarrow \underbrace{\begin{bmatrix} 3 & 0 \\ -2 & \frac{5}{3} \end{bmatrix}}_{\mathbf{L}}\underbrace{\begin{bmatrix} 1 & -\frac{2}{3} \\ 0 & 1 \end{bmatrix}}_{\mathbf{U}}$$

$$\begin{bmatrix} V_1^\circ \\ V_2^\circ \end{bmatrix}=\begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix}\leftarrow$$

(2) Construct $\mathbf{B}=\dfrac{\partial \mathbf{b}}{\partial \mathbf{g}}-\underbrace{\left[\dfrac{\partial \mathbf{A}}{\partial g_1}\mathbf{x}^\circ \cdots \dfrac{\partial \mathbf{A}}{\partial g_3}\mathbf{x}^\circ\right]}_{0}$ **Constructed from the stamp connectivity**

$$\frac{\partial \mathbf{A}}{\partial g_1}\mathbf{x}^\circ=\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix}=\begin{bmatrix} 3/5 \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{A}}{\partial g_2}\mathbf{x}^\circ=\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix}=\begin{bmatrix} 1/5 \\ -1/5 \end{bmatrix}$$

$$\frac{\partial \mathbf{A}}{\partial g_3}\mathbf{x}^\circ=\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 3/5 \\ 2/5 \end{bmatrix}=\begin{bmatrix} 0 \\ 2/5 \end{bmatrix}$$

$$\mathbf{B}=-\underbrace{\begin{bmatrix} 3/5 & 1/5 & 0 \\ 0 & -1/5 & 2/5 \end{bmatrix}}_{k}\ \Big\updownarrow\ n$$

Example (cont.)

(3) Solve $A^T u = e$ (Adjoint system)

$$\underbrace{\begin{bmatrix} 1 & 0 \\ -2/3 & 1 \end{bmatrix}}_{\mathbf{U}^T} \underbrace{\begin{bmatrix} 3 & -2 \\ 0 & 5/3 \end{bmatrix}}_{\mathbf{L}^T} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\mathbf{e}}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 2/5 \\ 3/5 \end{bmatrix} \leftarrow$$

$$(4)\ \frac{\partial v_{out}}{\partial \mathbf{g}} = \mathbf{u}^T \mathbf{B}$$

$$= -\begin{bmatrix} \frac{2}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} 3/5 & 1/5 & 0 \\ 0 & -1/5 & 2/5 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{6}{25} & \frac{1}{25} & -\frac{6}{25} \end{bmatrix}$$

Nonlinear Performance Function

- If the performance function is nonlinear $\phi = f(\mathbf{x}, \mathbf{p})$ (the system is still linear), then

$$\frac{\partial \phi}{\partial \mathbf{p}} = \frac{\partial f}{\partial \mathbf{p}} + \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}$$

$\frac{\partial f}{\partial \mathbf{p}}$ and $\mathbf{e}^{\mathbf{T}} = \frac{\partial f}{\partial \mathbf{x}}$ are evaluated at $\mathbf{x}^o$ and $\mathbf{p}^o$.

$\mathbf{e}^{\mathbf{T}} \left[\frac{\partial \mathbf{x}}{\partial \mathbf{p}} \right]$ is computed by the small-change sensitivity algorithm.

Partitioning and Block Decomposition

$$\mathrm{Ax} = \begin{bmatrix} \mathrm{A}_{11} & \mathrm{A}_{12} & \dots & \mathrm{A}_{1p} \\ \mathrm{A}_{21} & \mathrm{A}_{22} & \dots & \mathrm{A}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{A}_{p1} & \mathrm{A}_{p2} & \dots & \mathrm{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_p \end{bmatrix} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_p \end{bmatrix}$$

Partitioning and Block Decomposition

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{B}_{11} \\ \mathbf{C}_{11} & \mathbf{D}_{11} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{D}_{11}^{(1)} \end{bmatrix}$$

where $\mathbf{U}_{12} = \mathbf{L}_{11}^{-1}\mathbf{B}_{11}$, $\mathbf{L}_{21} = \mathbf{C}_{11}\mathbf{U}_{11}^{-1}$, and $\mathbf{D}_{11}^{(1)} = \mathbf{D}_{11} - \mathbf{L}_{21}\mathbf{U}_{12}$

Partitioning and Block Decomposition

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{C}_{11} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{D}_{11}^{(1)} \end{bmatrix}$$

$$\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{U}_{11},\ \mathbf{U}_{12} = \mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{B}_{11},$$

$$\mathbf{D}_{11}^{(1)} = \mathbf{D}_{11} - \mathbf{C}_{11}\mathbf{U}_{12} = \mathbf{D}_{11} - \mathbf{C}_{11}(\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{B}_{11})$$

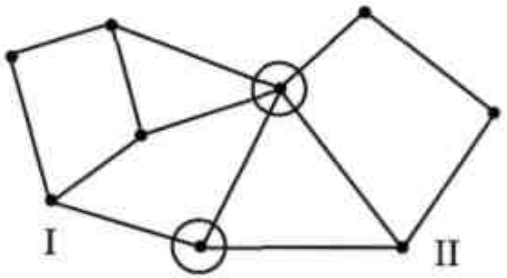
$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{B}_{11} \\ \mathbf{0} & \mathbf{D}_{11}^{(1)} \end{bmatrix}$$

$$\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{U}_{11},\ \mathbf{L}_{21} = \mathbf{C}_{11}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1},$$

$$\mathbf{D}_{11}^{(1)} = \mathbf{D}_{11} - \mathbf{L}_{21}\mathbf{B}_{11} = \mathbf{D}_{11} - (\mathbf{C}_{11}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1})\mathbf{B}_{11}$$

Bordered-Block Diagonal Form

Node Tearing:

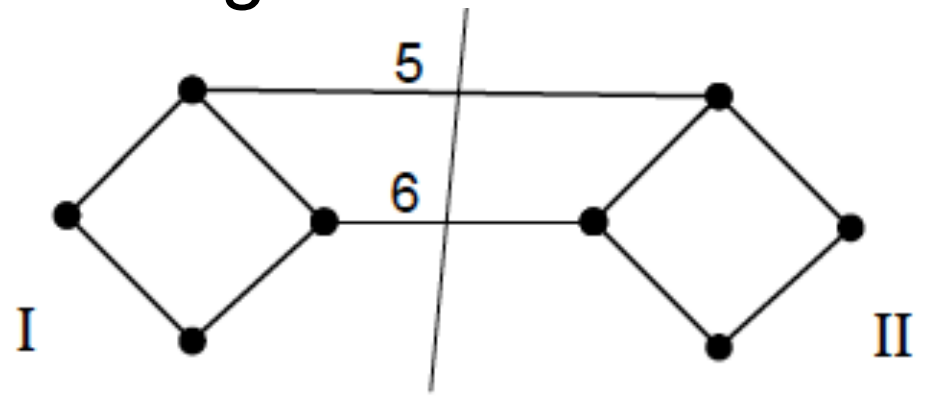


$$\left[\begin{array}{cc:c} \mathbf{A}_{11} & \mathbf{O} & \mathbf{A}_{1t} \\ \mathbf{O} & \mathbf{A}_{22} & \mathbf{A}_{2t} \\ \hdashline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{array}\right]\left[\begin{array}{c} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_t \end{array}\right]=\left[\begin{array}{c} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{array}\right]$$

In general:

$$\left[\begin{array}{ccc:c} \mathbf{A}_{11} & & \mathbf{O} & \mathbf{A}_{1t} \\ & \mathbf{A}_{22} & \vdots & \mathbf{A}_{2t} \\ \mathbf{O} & & \mathbf{A}_{pp} & \vdots \\ \hdashline \mathbf{A}_{t1} & \mathbf{A}_{t2} & \cdots & \mathbf{A}_{tt} \end{array}\right]\left[\begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \\ \mathbf{x}_t \end{array}\right]=\left[\begin{array}{c} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_t \end{array}\right]$$

Branch Tearing



$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \vdots & \mathbf{A}_{1t} \\ \mathbf{0} & \mathbf{A}_{22} & \vdots & \mathbf{A}_{2t} \\ \cdots & \cdots & \cdot & \cdots \\ \mathbf{A}_{t1}^{T} & \mathbf{A}_{t2}^{T} & \vdots & -\mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \cdots \\ \mathbf{i}_t \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \cdots \\ \mathbf{b}_t \end{bmatrix}$$

In most cases, the submatrices $\mathbf{A}_{1t}$ and $\mathbf{A}_{2t}$ contain +1, -1, and 0 only; $\mathbf{R}$ the resistance or impedance matrix of the tearing branches.

Floating Subcircuit

$$\begin{bmatrix} \mathbf{A}_{ii} & \vdots & \mathbf{a}_{ir} & \mathbf{A}_{\mathrm{it}} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{a}_{ri}^{\mathrm{T}} & \vdots & a_{rri} & \mathbf{a}_{rti}^{\mathrm{T}} \\ \mathbf{A}_{ti} & \vdots & \mathbf{a}_{tri} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \cdots \\ v_{ri} \\ \mathbf{i}_{ti} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_i \\ \cdots \\ b_{ri} \\ \mathbf{b}_{ti} \end{bmatrix}$$

Bordered-Block Diagonal Form

$$\left[\begin{array}{ccc:c} \mathbf{A}_{11} & & \mathbf{O} & \mathbf{A}_{1t} \\ & \mathbf{A}_{22} & \vdots & \mathbf{A}_{2t} \\ \mathbf{O} & & \mathbf{A}_{pp} & \vdots \\ \hdashline \mathbf{A}_{t1} & \mathbf{A}_{t2} & \cdots & \mathbf{A}_{tt} \end{array}\right]\left[\begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \\ \hdashline \mathbf{x}_t \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \hdashline \mathbf{b}_t \end{array}\right]$$

Solution of BBD Equations: Subcircuit Eqns. Factorization and Forward Subst

$$\begin{bmatrix} \mathbf{A}_{ii} & \mathbf{A}_{it} \\ \mathbf{A}_{ti} & \mathbf{A}_{tti} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{it} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_i \\ \mathbf{b}_{ti} \end{bmatrix}$$

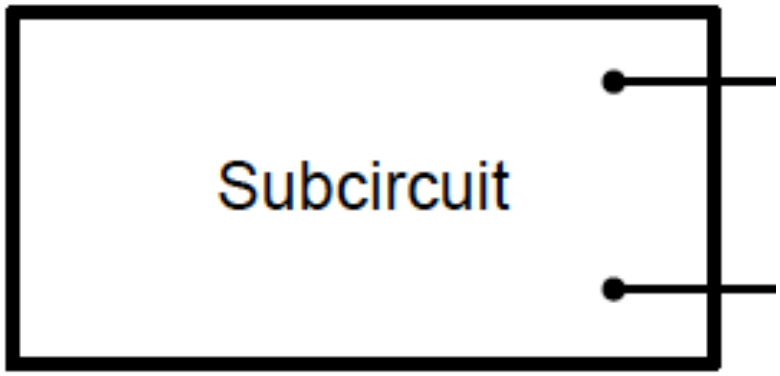
$$\begin{bmatrix} \mathbf{U}_{ii} & \mathbf{L}_{ii}^{-1}\mathbf{A}_{it} \\ \mathbf{0} & \tilde{\mathbf{A}}_{tti} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{it} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{ii}^{-1}\mathbf{b}_i \\ \tilde{\mathbf{b}}_{it} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{A}_{it} \\ \mathbf{0} & \tilde{\mathbf{A}}_{tti} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{it} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{b}_i \\ \tilde{\mathbf{b}}_{it} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{L}_{ii}\mathbf{U}_{ii} & \mathbf{A}_{it} \\ \mathbf{0} & \tilde{\mathbf{A}}_{tti} \end{bmatrix} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{it} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_i \\ \tilde{\mathbf{b}}_{it} \end{bmatrix}$$

where $\tilde{\mathbf{A}}_{tti} = \mathbf{A}_{tti} - (\mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1})\mathbf{A}_{it}$, $\tilde{\mathbf{b}}_{it} = \mathbf{b}_{it} - (\mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{b}_i)$

Subcircuit Equations



$$(\mathbf{A}_{tti} - \mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{A}_{it})\mathbf{x}_{it} = \mathbf{b}_{it} - (\mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1})\mathbf{b}_i$$

Terminal representation of the subcircuit at the tearing points

Solution of BBD Equations: Alg. I

Algorithm 4.8.1: *Solution Algorithm I of* $\mathbf{Ax} = \mathbf{b}$ *partitioned matrix in Bordered-Block Diagonal form*

{Subcircuit equations}

1. for $i = 1 : p$
2. while $\mathbf{A}_{ii}$ is nonsingular
3. factorize $\mathbf{A}_{ii} = \mathbf{L}_{ii}\mathbf{U}_{ii}$
4. $\mathbf{A}_{it} = \mathbf{L}_{ii}^{-1}\mathbf{A}_{it}$
5. $\mathbf{A}_{ti} = \mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}$
6. $\mathbf{A}_{tti} = \mathbf{A}_{tti} - \mathbf{A}_{ti}\mathbf{A}_{it}$
7. $\mathbf{b}_i = \mathbf{L}_{ii}^{-1}\mathbf{b}_i$
8. $\mathbf{b}_{ti} = \mathbf{b}_{ti} - \mathbf{A}_{ti}\mathbf{b}_i$
9. end

Solution of BBD Equations: Alg. I (cont.)

{Interconnect circuit equations}

10. for $i = 1 : p$
11. $\quad \mathbf{A}_{tt} = \mathbf{A}_{tt} + \mathbf{A}_{tti}$
12. $\quad \mathbf{b}_t = \mathbf{b}_t + \mathbf{b}_{ti}$
13. end
14. factorize $\mathbf{A}_{tt} = \mathbf{L}_{tt}\mathbf{U}_{tt}$
15. $\mathbf{b}_t = \mathbf{L}_{tt}^{-1}\mathbf{b}_t$
16. $\mathbf{b}_t = \mathbf{U}_{tt}^{-1}\mathbf{b}_t$

{Subcircuit backward substitution}

17. for $i = 1 : p$
18. $\quad \mathbf{b}_i = \mathbf{U}_{ii}^{-1}(\mathbf{b}_i - \mathbf{A}_{it}\mathbf{b}_t)$
19. end

Solution of BBD Equations: Alg. II

Algorithm 4.8.2: *Solution Algorithm II of* $\mathbf{Ax} = \mathbf{b}$ *partitioned matrix in Bordered-Block Diagonal form*

{Subcircuit equations}

1. for $i = 1 : p$
2. while $\mathbf{A}_{ii}$ is nonsingular
3. factorize $\mathbf{A}_{ii} = \mathbf{L}_{ii}\mathbf{U}_{ii}$
4. $\mathbf{A}_{it} = \mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{A}_{it}$
5. $\mathbf{A}_{tti} = \mathbf{A}_{tti} - \mathbf{A}_{ti}\mathbf{A}_{it}$
6. $\mathbf{b}_i = \mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}\mathbf{b}_i$
7. $\mathbf{b}_{ti} = \mathbf{b}_{ti} - \mathbf{A}_{ti}\mathbf{b}_i$
8. end

Solution of BBD Equations: Alg. II (cont.)

{Interconnect circuit equations}

9. for $i = 1 : p$

10. $\mathbf{A}_{tt} = \mathbf{A}_{tt} + \mathbf{A}_{tti}$

11. $\mathbf{b}_t = \mathbf{b}_t + \mathbf{b}_{ti}$

12. end

13. factorize $\mathbf{A}_{tt} = \mathbf{L}_{tt}\mathbf{U}_{tt}$

14. $\mathbf{b}_t = \mathbf{L}_{tt}^{-1}\mathbf{b}_t$

15. $\mathbf{b}_t = \mathbf{U}_{tt}^{-1}\mathbf{b}_t$

{Subcircuit backward substitution}

16. for $i = 1 : p$

17. $\mathbf{b}_i = \mathbf{b}_i - \mathbf{A}_{it}\mathbf{b}_t$

19. end

Solution of BBD Equations: Alg. III

Algorithm 4.8.3: *Solution Algorithm III of* $\mathbf{Ax} = \mathbf{b}$ *partitioned matrix in Bordered-Block Diagonal form*

{Subcircuit equations}

1. for $i = 1 : p$
2. while $\mathbf{A}_{ii}$ is nonsingular
3. factorize $\mathbf{A}_{ii} = \mathbf{L}_{ii}\mathbf{U}_{ii}$
4. $\mathbf{A}_{ti} = \mathbf{A}_{ti}\mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}$
5. $\mathbf{A}_{tti} = \mathbf{A}_{tti} - \mathbf{A}_{ti}\mathbf{A}_{it}$
6. $\mathbf{b}_{ti} = \mathbf{b}_{ti} - \mathbf{A}_{ti}\mathbf{b}_{i}$
7. end

Solution of BBD Equations: Alg. III (cont.)

{Interconnect circuit equations}

8. for $i = 1 : p$
9. $\quad \mathbf{A}_{tt} = \mathbf{A}_{tt} + \mathbf{A}_{tti}$
10. $\quad \mathbf{b}_t = \mathbf{b}_t + \mathbf{b}_{ti}$
11. end
12. factorize $\mathbf{A}_{tt} = \mathbf{L}_{tt}\mathbf{U}_{tt}$
13. $\mathbf{b}_t = \mathbf{L}_{tt}^{-1}\mathbf{b}_t$
14. $\mathbf{b}_t = \mathbf{U}_{tt}^{-1}\mathbf{b}_t$

{Subcircuit backward substitution}

15. for $i = 1 : p$
16. $\quad \mathbf{b}_i = \mathbf{U}_{ii}^{-1}\mathbf{L}_{ii}^{-1}(\mathbf{b}_i - \mathbf{A}_{it}\mathbf{b}_t)$
17. end

Nested Decomposition

- Also known as:

  Nested Dissection

  Domain Decomposition