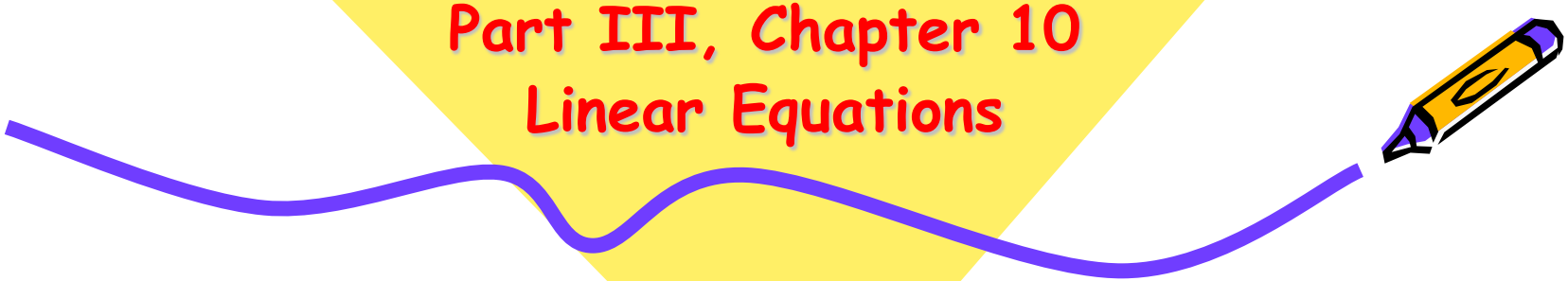
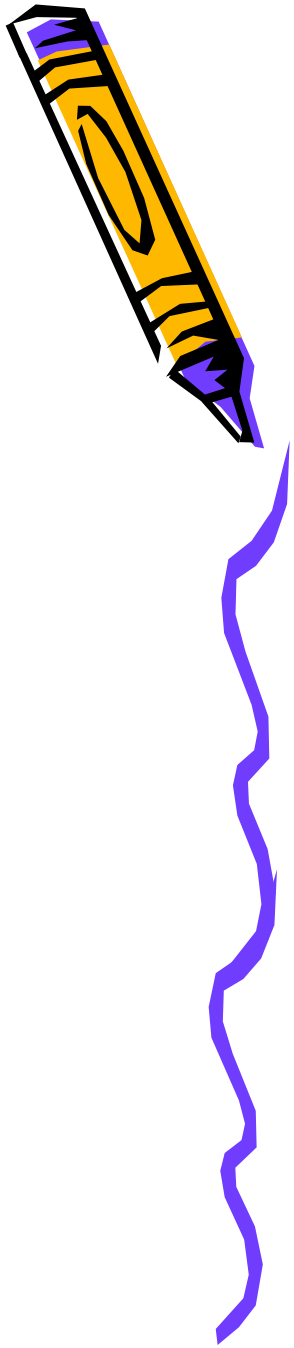
CPE 332
Computer Engineering Mathematics II

Week 12
Part III, Chapter 10
Linear Equations

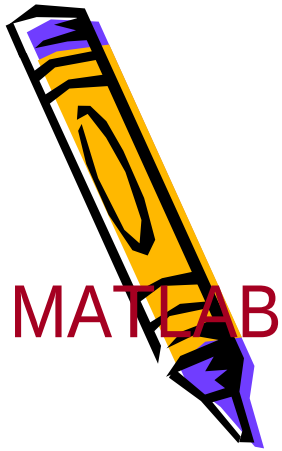


Today(Week 13) Topics



- Chapter 9 Linear Equations
  - Gauss Elimination
  - Gauss-Jordan
  - Gauss-Seidel
  - LU Decomposition
  - Crout Decomposition
- HW(9) Ch 9 Due Next Week

MATLAB Programming



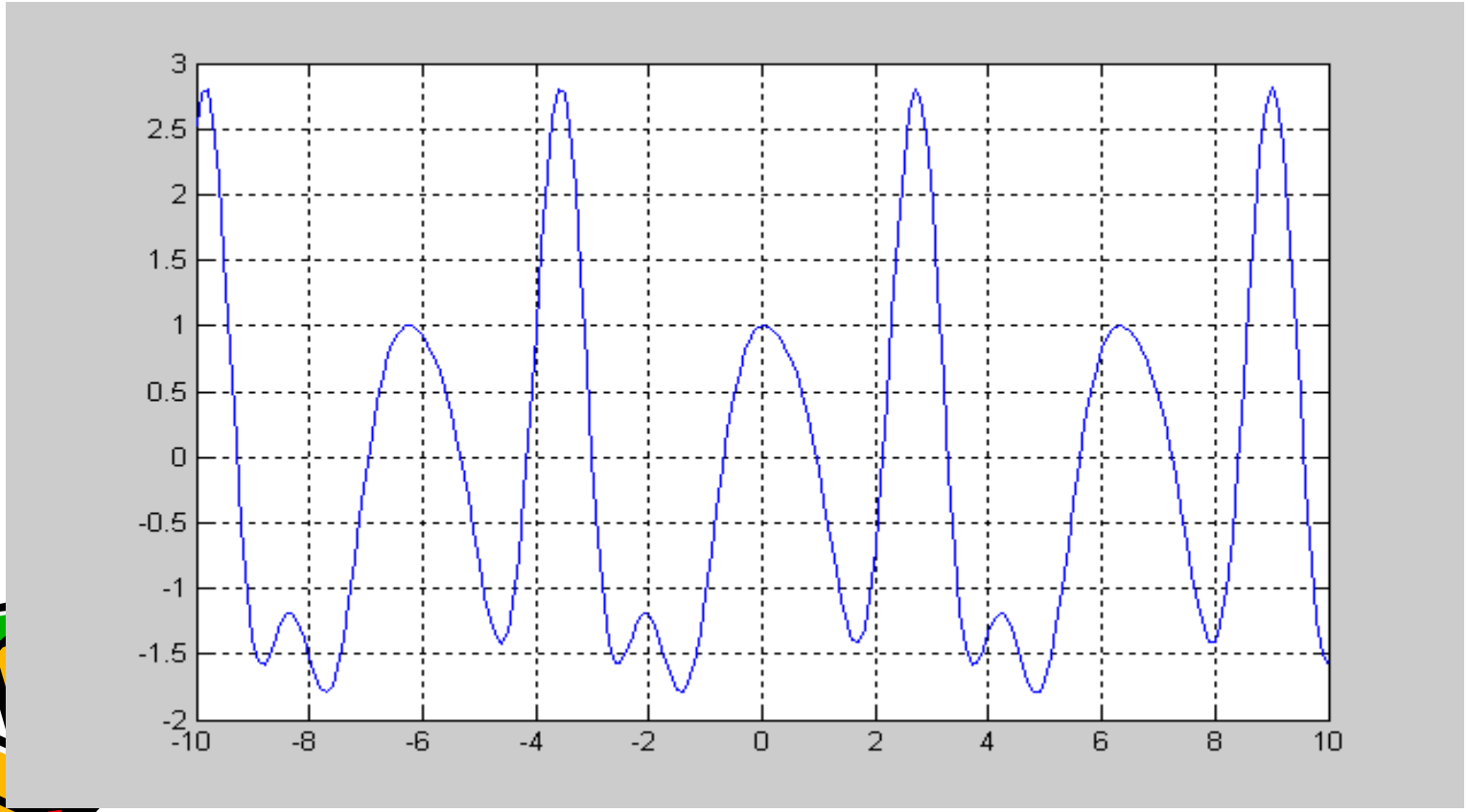
- เราสามารถเขียน Function การคำนวณโดยใช้ MATLAB Editor และบันทึกเป็น '.m' File
  - ขึ้นบันทัดแรกของ Function ด้วย

    function [List ของค่าที่ส่งคืน]=fname(List ของ Parameter)

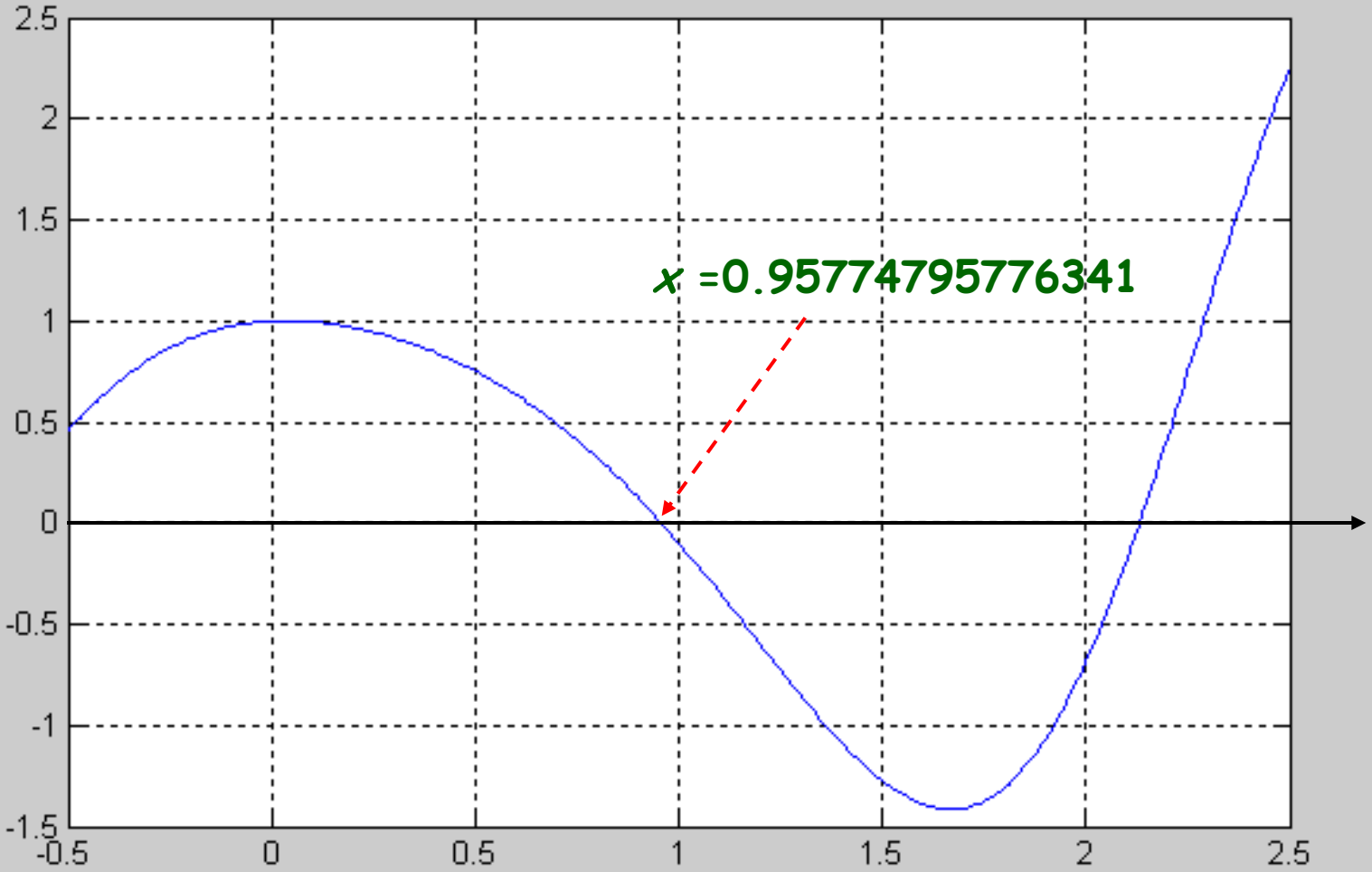
    function [x,y,z]=find123(a,b,c)
  - ภายใน Function สามารถใช้ Loop, Branch ได้เหมือนการเขียนโปรแกรม, สามารถกำหนด Local Variable ภายในได้เช่นกัน
  - อย่าลืมว่า พื้นฐาน Variable จะเป็น Matrix
- Function นี้สามารถเรียกใช้งานได้ใน MATLAB
- ดูรายละเอียดใน Tutorial 4-5 ของ MATLAB

Ex: หารากของ $f(x) = \sin 3x \cdot e^{-\cos x} + \cos 2x \cdot e^{-\sin x}$

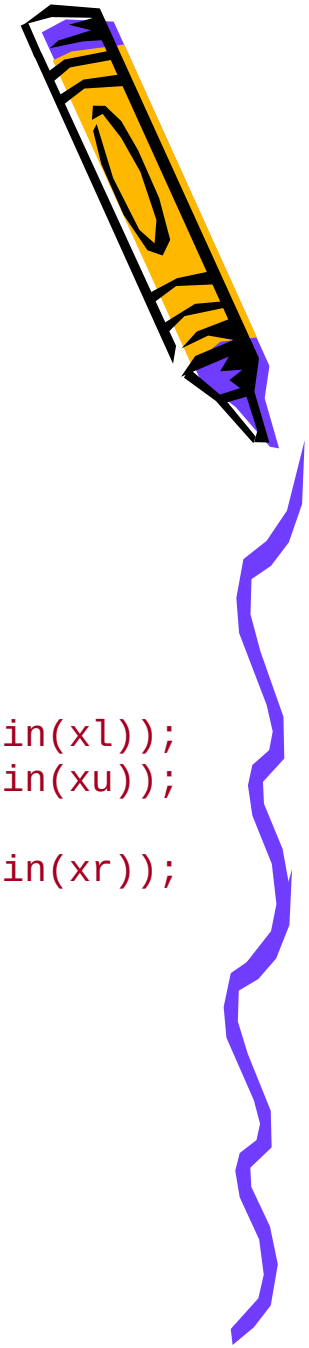
- `x=-10:.1:10;`
- `y=sin(3*x).*exp(-cos(x))+cos(2*x).*exp(-sin(x));`
- `plot(x,y)`



เราจะหาคำตอบในช่วง [0, 2]



MATLAB: Bisection Mtd.



```
function [x]=example91a(es)
% Calculate using Bisection Method between [0,2]
ea = inf;
xr = inf;
it=0;
xl=0;
xu=2;
while(ea > es)
    it = it+1;
    pxr=xr;
    fxl=sin(3*xl).*exp(-cos(xl))+cos(2*xl).*exp(-sin(xl));
    fxu=sin(3*xu).*exp(-cos(xu))+cos(2*xu).*exp(-sin(xu));
    xr=(xl+xu)/2;
    fxr=sin(3*xr).*exp(-cos(xr))+cos(2*xr).*exp(-sin(xr));
    ea = abs((xr-pxr)/xr)*100;
    x=[it xl fxl xu fxu xr fxr ea]
    if(fxl*fxr > 0.0)
        xl=xr;
    elseif (fxl*fxr < 0.0)
        xu=xr;
    else
        ea=0.0;
    end
end
```

Bisection Results:>> example91a(0.01)

	Iter	xl	fxl	xu	fxu	xr	fxr	ea
x =	1.0000	0	1.0000	2.0000	-0.6869	1.0000	-0.0972	Inf
x =	2.0000	0	1.0000	1.0000	-0.0972	0.5000	0.7493	100.0000
x =	3.0000	0.5000	0.7493	1.0000	-0.0972	0.7500	0.4101	33.3333
x =	4.0000	0.7500	0.4101	1.0000	-0.0972	0.8750	0.1774	14.2857
x =	5.0000	0.8750	0.1774	1.0000	-0.0972	0.9375	0.0451	6.6667
x =	6.0000	0.9375	0.0451	1.0000	-0.0972	0.9688	-0.0249	3.2258
x =	7.0000	0.9375	0.0451	0.9688	-0.0249	0.9531	0.0104	1.6393
x =	8.0000	0.9531	0.0104	0.9688	-0.0249	0.9609	-0.0072	0.8130
x =	9.0000	0.9531	0.0104	0.9609	-0.0072	0.9570	0.0016	0.4082
x =	10.0000	0.9570	0.0016	0.9609	-0.0072	0.9590	-0.0028	0.2037
x =	11.0000	0.9570	0.0016	0.9590	-0.0028	0.9580	-0.0006	0.1019
x =	12.0000	0.9570	0.0016	0.9580	-0.0006	0.9575	0.0005	0.0510
x =	13.0000	0.9575	0.0005	0.9580	-0.0006	0.9578	-0.0000	0.0255
x =	14.0000	0.9575	0.0005	0.9578	-0.0000	0.9576	0.0002	0.0127
x =	15.0000	0.9576	0.0002	0.9578	-0.0000	0.9577	0.0001	0.0064

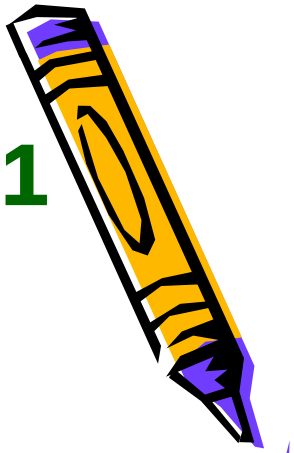
ans =

15.00000000000000 0.95776367187500 -0.00003535871565 0.95770263671875 0.00023929892750 **0.95770263671875 0.00010197464576 0.00637308010962**

x =0.95774795776341

True error = 0.004732%

Other Results: xt= **0.957747957763 41**



- Es = 0.01%
  - It = 15;xr=0.957702636718 75
  - Ea=0.006373%, et = 0.004732%
- Es = Es = 0.001%
  - It = 18;xr=0.957740783691 41
  - Ea=0.0007966%, et = 0.0007491%
- Es = 0.0001%
  - It = 21;xr=0.957747459411 62
  - Ea=0.00009957%, et = 0.00005203%
- Es = 0.000001%
  - It = 28;xr=0.957747958600 52
  - Ea=0.0000007779%, et = 8.740e-008%

MATLAB: False-Position

```
function [x]=example91b(es)
% Calculate using False-Position Method between [0,2]
ea = inf;
xr = inf;
it=0;
xl=0;
xu=2;
while(ea > es)
    it = it+1;
    pxr=xr;
    fxl=sin(3*xl).*exp(-cos(xl))+cos(2*xl).*exp(-sin(xl));
    fxu=sin(3*xu).*exp(-cos(xu))+cos(2*xu).*exp(-sin(xu));
    % xr=(xl+xu)/2;
    xr=xu-((fxu*(xl-xu))/(fxl-fxu));
    fxr=sin(3*xr).*exp(-cos(xr))+cos(2*xr).*exp(-sin(xr));
    ea = abs((xr-pxr)/xr)*100;
    x=[it xl fxl xu fxu xr fxr ea]
    if(fxl*fxr > 0.0)
        xl=xr;
    elseif (fxl*fxr < 0.0)
        xu=xr;
    else
        ea=0.0;
    end
end
```

FP Results:>> example91b(0.01)

	Iter	xl	fxl	xu	fxu	xr	fxr	ea
x =	1.0000	0	1.0000	2.0000	-0.6869	1.1856	-0.5611	Inf
x =	2.0000	0	1.0000	1.1856	-0.5611	0.7595	0.3940	56.1096
x =	3.0000	0.7595	0.3940	1.1856	-0.5611	0.9353	0.0500	18.7964
x =	4.0000	0.9353	0.0500	1.1856	-0.5611	0.9557	0.0045	2.1423
x =	5.0000	0.9557	0.0045	1.1856	-0.5611	0.9576	0.0004	0.1922
x =	6.0000	0.9576	0.0004	1.1856	-0.5611	0.9577	0.0000	0.0166
x =	7.0000	0.9577	0.0000	1.1856	-0.5611	0.9577	0.0000	0.0014

ans =

7.00000000000000 0.95773289766706 0.00003388653487 1.18559512875289
-0.56109590391892 0.95774665822935 0.00000292408716 0.00143676432376

x =0.95774795776341

True error = 0.0001357 %

Other Results: x_t= **0.957747957763 41**

- สีน้ำเงินได้จาก Bisection Method
- สีเขียวได้จาก False-Position Method
- Es = 0.01%
  - It = 15;xr=0.95770263671875 Ea=0.006373%, et = 0.004732%
  - It = 7;xr=0.957746658229 35 Ea=0.001437%, et = 0.0001357%
- Es = Es = 0.001%
  - It = 18;xr=0.957740783691 41 Ea=0.0007966%, et = 0.0007491%
  - It = 8;xr=0.957747845629 42 Ea=0.0001240%, et = 0.00001171%
- Es = 0.0001%
  - It = 21;xr=0.957747459411 62 Ea=0.00009957%, et = 0.00005203%
  - It = 9;xr=0.957747948087 63 Ea=0.00001070%, et = 0.000001010%
- Es = 0.000001%
  - It = 28;xr=0.957747958600 52 Ea=0.0000007779%, et = 8.740e-008%
  - It = 10;xr=0.957747956928 51 Ea=0.0000009231%, et = 8.717e-008%

Newton-Ralphson Method

$$f(x) = \sin 3x \cdot e^{-\cos x} + \cos 2x \cdot e^{-\sin x}$$

$$f'(x) = \sin 3x \cdot d\frac{e^{-\cos x}}{dx} + e^{-\cos x} d\frac{\sin 3x}{dx} + \cos 2x \cdot d\frac{e^{-\sin x}}{dx} + e^{-\sin x} d\frac{\cos 2x}{dx}$$

$$= \sin 3x \cdot e^{-\cos x} \cdot \sin x + e^{-\cos x} \cdot 3\cos 3x + \cos 2x \cdot e^{-\sin x}(-\cos x) + e^{-\sin x} \cdot (-2\sin 2x)$$

$$= e^{-\cos x}[\sin 3x \sin x + 3\cos 3x] - e^{-\sin x}[\cos 2x \cos x + 2\sin 2x]$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{\sin 3x_i \cdot e^{-\cos x_i} + \cos 2x_i \cdot e^{-\sin x_i}}{e^{-\cos x_i}[\sin 3x_i \sin x_i + 3\cos 3x_i] - e^{-\sin x_i}[\cos 2x_i \cos x_i + 2\sin 2x_i]}$$

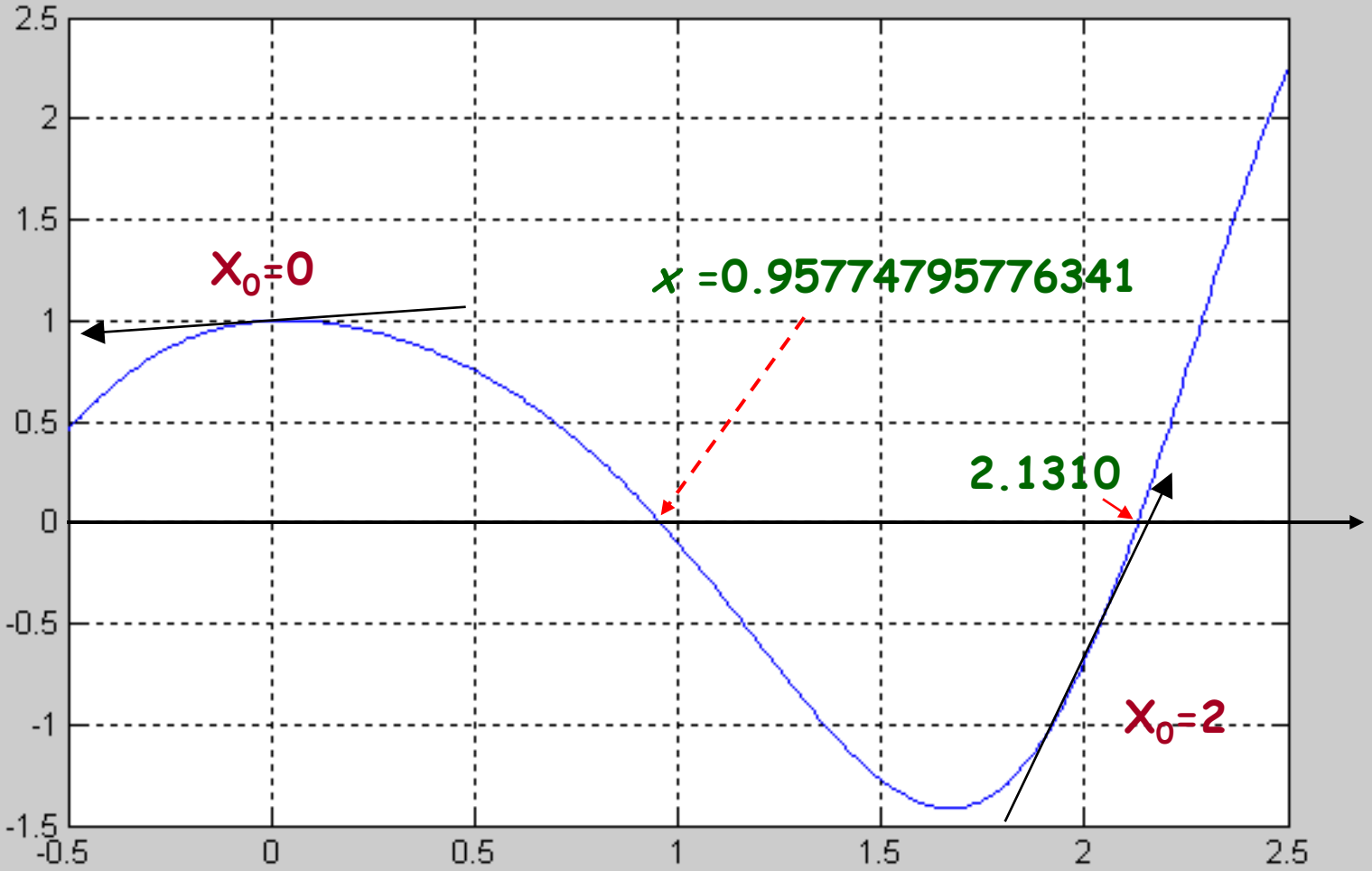
MATLAB Program:

```
function [x]=example91c(es,x0)

% Calculate solution using Newton-Ralphson, x0=initial;

it=0;
xi=x0;
ea=inf;
while (ea > es)
    it = it+1;
    fxi=sin(3*xi)*exp(-cos(xi))+cos(2*xi)*exp(-sin(xi));
    dfxi=exp(-cos(xi))*(sin(3*xi)*sin(xi)+3*cos(3*xi))...
        -exp(-sin(xi))*(cos(2*xi)*cos(xi)+2*sin(2*xi));
    pxi=xi;
    xi=pxi-fxi/dfxi;
    ea=abs((xi-pxi)/xi)*100;
    x=[it pxi fxi dfxi xi ea]
end
```

Result: ea=0.01, x_0=?

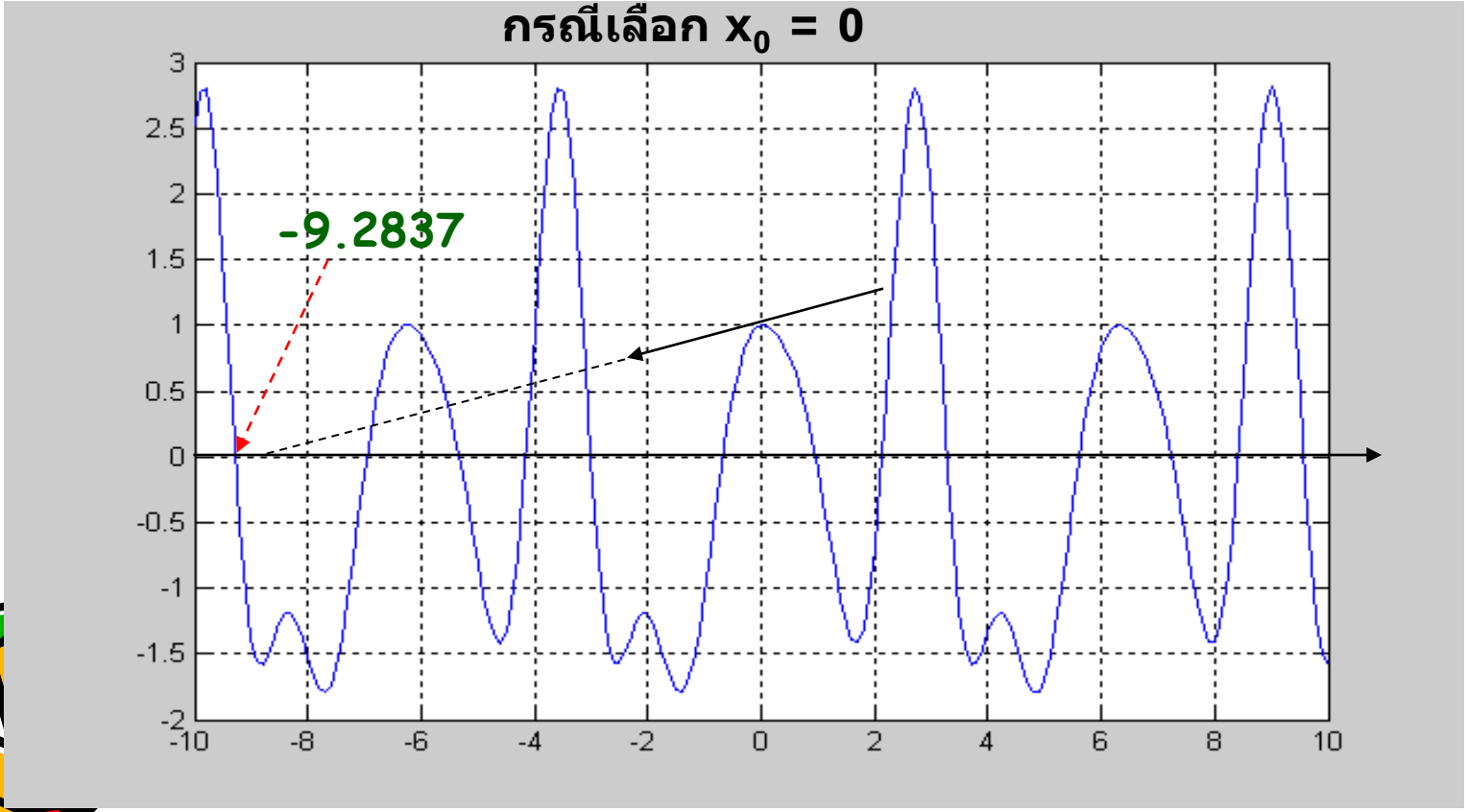
- X_0=0 โปรแกรมจะ Converge เข้าสู่จุดอื่นด้านซ้าย
- X_0=2 โปรแกรมจะ Converge เข้าสู่จุดอื่นด้านขวา
- ดูรูป
- ถ้า x_0 = 0.5 หรือ 1.5 โปรแกรมจะ Converge เข้าจุดที่ต้องการอย่างรวดเร็วมาก
- เป็นไปได้ที่เราเลือกจุดที่โปรแกรมไม่ Converge
- เราอาจจะใช้ Bisection Method ก่อนเพื่อหาจุด x_0 ที่ดี จากนั้นต่อด้วย Newton-Ralphson เพื่อให้ได้คำตอบอย่างรวดเร็ว

เราจะหาคำตอบในช่วง [0, 2]



Ex: หารากของ $f(x) = \sin 3x \cdot e^{-\cos x} + \cos 2x \cdot e^{-\sin x}$

- `x=-10:.1:10;`
- `y=sin(3*x).*exp(-cos(x))+cos(2*x).*exp(-sin(x));`
- `plot(x,y)`



Result: x_0=0.5, es = 0.01

	Iter	xi	fxi	dfxi	xi+1	ea
• x =	1.0000	0.5000	0.7493	-1.0485	1.2146	58.8351
• x =	2.0000	1.2146	-0.6362	-2.5824	0.9683	25.4413
• x =	3.0000	0.9683	-0.0238	-2.2755	0.9578	1.0939
• x =	4.0000	0.9578	-0.0001	-2.2502	0.9577	0.0061

- x = 4 0.95780651884294 -0.00013177280 -2.25024858111352 0.95774795962033 0.00611426231998

***x* =0.95774795776341**
True error = 0.0000001939 %

Result: x_0=0.5, es = 0.000001

	Iter	xi	fxi	dfxi	xi+1	ea
• x =	1.0000	0.5000	0.7493	-1.0485	1.2146	58.8351
• x =	2.0000	1.2146	-0.6362	-2.5824	0.9683	25.4413
• x =	3.0000	0.9683	-0.0238	-2.2755	0.9578	1.0939
• x =	4.0000	0.9578	-0.0001	-2.2502	0.9577	0.0061
• x =	5.0000	0.9577	-0.0000	-2.2501	0.9577	0.0000

- x = 5 0.957747959620 33 -0.000000004178 26 - 2.250105876352 98 0.957747957763 41 0.000001938834 9

x =0.957747957763 41

True error = 0.000000000000000 %

Compare : *xt* =0.957747957763 41

- Es = 0.01%
  - It = 15;xr=0.95770263671875 Ea=0.006373%, et = 0.004732%
  - It = 7;xr=0.957746658229 35 Ea=0.001437%, et = 0.0001357%
  - It = 4;xi= 0.957747959620 33 Ea=0.006114%, et = 0.0000001939 %
- Es = Es = 0.001%
  - It = 18;xr=0.957740783691 41 Ea=0.0007966%, et = 0.0007491%
  - It = 8;xr=0.957747845629 42 Ea=0.0001240%, et = 0.00001171%
  - It = 5;xi= 0.957747957763 41 Ea=0.0000001939%, et < 1.0e-15 %
- Es = 0.0001%
  - It = 21;xr=0.957747459411 62 Ea=0.00009957%, et = 0.00005203%
  - It = 9;xr=0.957747948087 63 Ea=0.00001070%, et = 0.000001010%
  - It = 5;xi= 0.957747957763 41 Ea=0.0000001939%, et < 1.0e-15 %
- Es = 0.000001%
  - It = 28;xr=0.957747958600 52 Ea=0.0000007779%, et = 8.740e-008%
  - It = 10;xr=0.957747956928 51 Ea=0.0000009231%, et = 8.717e-008%
  - It = 5;xi= 0.957747957763 41 Ea=0.0000001939%, et < 1.0e-15 %

เพียง 5 iteration วิธีของ Newton-Ralphson
ให้ Error น้อยจน Double Precision วัดไม่ได้
แต่ข้อเสียคือจุด x_0 จะต้องเลือกให้ดี

Chapter 10: System of Linear Eq.

- จะ Limit อยู่ที่สมการ $AX=B$ โดย A เป็น Square Matrix
  - N สมการ N Unknown
  - จะมีคำตอบที่ Unique
  - คำตอบจะมีได้ต่อเมื่อ A ไม่เป็น Singular
    - Determinant ไม่เท่ากับ 0
    - A หา Inverse ได้ และ $X = A^{-1}B$
  - ในกรณีที่ Determinant A ใกล้ศูนย์ แต่ไม่ใช่ศูนย์ คำตอบจะ Sensitive กับ Error การคำนวณเมื่อมีการปัดเศษจะต้องระวัง
    - กรณีนี้ เราเรียกว่ามันเป็น 'Ill-Conditioned System'

System of Linear Equations

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = c_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = c_2 \\
&\qquad\vdots \qquad\quad \vdots \qquad\quad \vdots \\
&a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = c_n
\end{aligned}
\quad = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \mathbf{AX} = \mathbf{C}
$$

Krammer's Rule

$$x_1 = \frac{\begin{vmatrix} c_1 & a_{12} & a_{13} \\ c_2 & a_{22} & a_{23} \\ c_3 & a_{32} & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

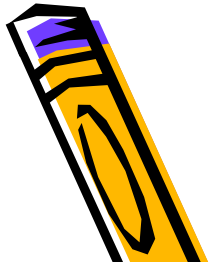
Solution ของ $AX=C$

- $A^{-1}AX=A^{-1}C$
- $X=A^{-1}C$
- Inverse หาได้ยาก แม้จะใช้ Computer คำนวณ เพราะเป็น $O(n^4)$

Solution by Elimination

วิธีการ Elimination ของ Unknown ยังสามารถนำมาใช้ได้ในกรณีที่เรามี Unknown มากกว่า 3 ตัว ด้วยการหา Factor ร่วมของคู่ของสมการ และกำจัด หนึ่ง Unknown ออกไป ทำให้เราเหลือระบบที่มี $n-1$ Unknown และ $n-1$ สมการ และเราก็สามารถทำเป็น Iteration(Loop) ต่อไปจนเหลือแค่ Unknown เดียว ซึ่งเราจะได้คำตอบสำหรับ Unknown นั้น จากนั้นเราสามารถที่จะแทนค่าย้อนกลับหา Unknown ตัวที่สอง สาม จนถึงตัวสุดท้าย ซึ่งวิธีการเช่นนี้ แม้ว่าจะใช้การคำนวณหลายขั้นตอน แต่เป็นวิธีการที่เหมาะสมในการเขียนโปรแกรมคอมพิวเตอร์ เพราะว่าจะได้ Source Code ที่สั้น วิธีการดังกล่าวก็คือ Gauss Elimination Method ที่จะกล่าวในหัวข้อต่อไป

Gauss Elimination



8.2.1 หลักการของ Gauss Elimination

1. ใน Elimination Step จาก AX=C เราพยายามทำให้ Matrix A อยู่ในรูป Upper Diagonal Matrix ด้วยขบวนการ Elimination คือการบวกและลบแต่ละแถวเข้าด้วยกัน และค่า C จะถูกบวกลบตามไปด้วย

2. เมื่อ A เป็น Upper Diagonal แล้ว การแก้สมการสามารถทำได้ง่าย โดยเราหา x_n ก่อนในแถวสุดท้ายของสมการ จากนั้นนำ x_n ที่หาได้มาแทนค่า เพื่อหา x_{n-1} ในแถวรองสุดท้าย เนื่องจากเป็นการแทนค่าเพื่อหา Unknown ย้อนหลัง เราจึงเรียก Back-Substitution

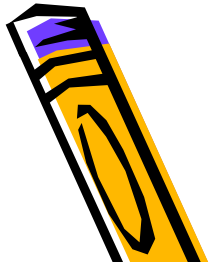
Gauss Elimination

- จาก

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= c_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= c_2 \\
\vdots \qquad\quad & \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= c_n
\end{aligned}
$$

$$\mathbf{AX} = \mathbf{C}$$

Gauss Elimination



8.2.1 หลักการของ Gauss Elimination

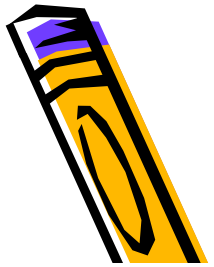
$$\mathbf{AX} = \mathbf{C}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \cdots & \hat{a}_{1,n} \\ a_{2,1} & \hat{a}_{2,2} & \cdots & \hat{a}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{a}_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \vdots \\ \hat{c}_n \end{bmatrix}$$

Gauss Elimination



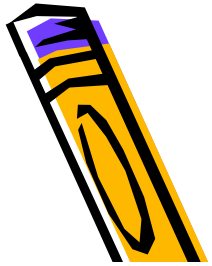
8.2.1 หลักการของ Gauss Elimination

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & a'_{3,2} & a'_{3,3} & a'_{3,4} & a'_{3,5} \\ 0 & a'_{4,2} & a'_{4,3} & a'_{4,4} & a'_{4,5} \\ 0 & a'_{5,2} & a'_{5,3} & a'_{5,4} & a'_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c'_3 \\ c'_4 \\ c'_5 \end{bmatrix}$$

Gauss Elimination



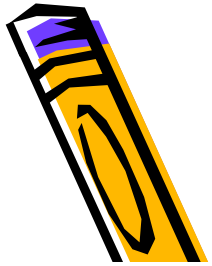
8.2.1 หลักการของ Gauss Elimination

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & a'_{3,2} & a'_{3,3} & a'_{3,4} & a'_{3,5} \\ 0 & a'_{4,2} & a'_{4,3} & a'_{4,4} & a'_{4,5} \\ 0 & a'_{5,2} & a'_{5,3} & a'_{5,4} & a'_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c'_3 \\ c'_4 \\ c'_5 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & a''_{4,3} & a''_{4,4} & a''_{4,5} \\ 0 & 0 & a''_{5,3} & a''_{5,4} & a''_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c''_4 \\ c''_5 \end{bmatrix}$$

Gauss Elimination



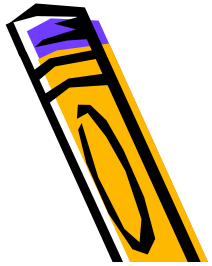
8.2.1 หลักการของ Gauss Elimination

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & a''_{4,3} & a''_{4,4} & a''_{4,5} \\ 0 & 0 & a''_{5,3} & a''_{5,4} & a''_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c''_4 \\ c''_5 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & 0 & a'''_{4,4} & a'''_{4,5} \\ 0 & 0 & 0 & a'''_{5,4} & a'''_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c'''_4 \\ c'''_5 \end{bmatrix}$$

Gauss Elimination



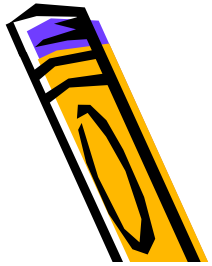
8.2.1 หลักการของ Gauss Elimination

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & 0 & a'''_{4,4} & a'''_{4,5} \\ 0 & 0 & 0 & a'''_{5,4} & a'''_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c'''_4 \\ c'''_5 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & 0 & a^{(3)}_{4,4} & a^{(3)}_{4,5} \\ 0 & 0 & 0 & 0 & a^{(4)}_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c^{(3)}_4 \\ c^{(4)}_5 \end{bmatrix}$$

Back Substitution



8.2.1 หลักการของ Gauss Elimination

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} & a'_{2,5} \\ 0 & 0 & a''_{3,3} & a''_{3,4} & a''_{3,5} \\ 0 & 0 & 0 & a^{(3)}_{4,4} & a^{(3)}_{4,5} \\ 0 & 0 & 0 & 0 & a^{(4)}_{5,5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \\ c^{(3)}_4 \\ c^{(4)}_5 \end{bmatrix}$$

$$\Downarrow$$

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 + a_{1,5}x_5 = c_1$$

$$a'_{2,2}x_2 + a'_{2,3}x_3 + a'_{2,4}x_4 + a'_{2,5}x_5 = c'_2$$

$$a''_{3,3}x_3 + a''_{3,4}x_4 + a''_{3,5}x_5 = c''_3$$

$$a^{(3)}_{4,4}x_4 + a^{(3)}_{4,5}x_5 = c^{(3)}_4$$

$$a^{(4)}_{5,5}x_5 = c^{(4)}_5$$

Back Substitution

$$x_i = \frac{c_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}}$$

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + a_{1,4}x_4 + a_{1,5}x_5 = c_1$$

$$a'_{2,2}x_2 + a'_{2,3}x_3 + a'_{2,4}x_4 + a'_{2,5}x_5 = c'_2$$

$$a''_{3,3}x_3 + a''_{3,4}x_4 + a''_{3,5}x_5 = c''_3$$

$$a_{4,4}^{(3)}x_4 + a_{4,5}^{(3)}x_5 = c_4^{(3)}$$

$$a_{5,5}^{(4)}x_5 = c_5^{(4)}$$

$$\Downarrow$$

$$x_5 = \frac{c_5^{(4)}}{a_{5,5}^{(4)}}$$

$$x_4 = \frac{c_4^{(3)} - a_{4,5}^{(3)}x_5}{a_{4,4}^{(3)}}$$

$$x_3 = \frac{c''_3 - a''_{3,4}x_4 - a''_{3,5}x_5}{a''_{3,3}}$$

$$x_2 = \frac{c'_2 - a'_{2,3}x_3 - a'_{2,4}x_4 - a'_{2,5}x_5}{a'_{2,2}}$$

$$x_1 = \frac{c_1 - a_{1,2}x_2 - a_{1,3}x_3 - a_{1,4}x_4 - a_{1,5}x_5}{a_{1,1}}$$

Gauss Elimination

การคูณสมการบรรทัดที่หนึ่ง ด้วยค่า a_{21}/a_{11} และได้

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}c_1$$

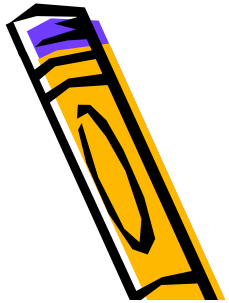
จากนั้นนำค่าที่ได้ หักลบออกจากสมการในบรรทัดที่สอง ทำให้ x_1 หายไป และได้สมการในบรรทัดที่สองใหม่เป็น

$$(a_{22} - \frac{a_{21}}{a_{11}}a_{12})x_2 + \cdots + (a_{2n} - \frac{a_{21}}{a_{11}}a_{1n})x_n = c_2 - \frac{a_{21}}{a_{11}}c_1$$

ด้วยการเปลี่ยน Variable สมการสามารถเขียนในรูปของ

$$a'_{22}x_2 + \cdots + a'_{2n}x_n = c'_2$$

Gauss Elimination



จากนั้นทำการกำจัด Unknown ตัวแรกของแถวที่ 3, 4 จนถึงแถวสุดท้าย โดยการคูณด้วย Factor $a_{31}/a_{11}, a_{41}/a_{11},....$ ในแถวแรก และนำมาหักออกจากแถวที่ต้องการ เราจะได้

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= c_1 \\
0 + a'_{22}x_2 + \cdots + a'_{2n}x_n &= c'_2 \\
\vdots \quad & \quad \vdots \quad \vdots \\
0 + a'_{n2}x_2 + \cdots + a'_{nn}x_n &= c'_n
\end{aligned}
$$

Gauss Elimination

ขบวนการสามารถกระทำซ้ำ โดยการกำจัด Unknown ที่เหลือทีละตัว โดยเริ่มจากแถวถัดไป (กำจัด x_1 เริ่มจากแถวที่สอง , กำจัด x_2 เริ่มจากแถวที่สาม จนถึงกำจัด x_{n-1} ออกจากแถวที่ n) สุดท้ายเราจะเหลือแค่ Upper Triangular Matrix

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= c_1 \\
0 + a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= c'_2 \\
0 + 0 + a''_{33}x_3 + \cdots + a''_{2n}x_n &= c''_3 \\
\vdots \quad \vdots \quad \vdots & \\
0 + 0 + \cdots + 0 + a_{nn}^{(n-1)}x_n &= c_n^{(n-1)}
\end{aligned}
$$

Gauss Elimination

ยกตัวอย่างกรณีของ 3 Unknown Equation จาก

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c'_2 \\ c''_3 \end{bmatrix}$$

Gauss Elimination

เมื่อได้ดังนี้แล้ว เราสามารถใช้ขบวนการ Back-Substitution แทนค่าย้อนกลับตั้งแต่ x_3 ดังนี้

$$x_3 = \frac{c''_3}{a''_{33}}$$

$$x_2 = \frac{c'_2 - a'_{23}\, x_3}{a'_{22}}$$

$$x_1 = \frac{c_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}$$

Gauss Elimination Alg

- Elimination by Forward Substitution

Forward Substitution Pseudo Code: (FORTRAN STYLE)

```
DOFOR k = 1 to n-1
   DOFOR i = k+1 to n
      factor = a(i,k)/a(k,k)
      DOFOR j = k+1 to n
         a(i,j) = a(i,j)-factor*a(k,j)
      ENDDO
      c(i) = c(i)-factor*c(k)
   ENDDO
ENDDO
```

Gauss Elimination Alg

- Back-Substitution

สำหรับขั้นตอนของ Back-Substitution เราเริ่มจาก

$$x_n = \frac{c_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

และเราสามารถพิสูจน์ได้ว่า Unknown ที่เหลือสามารถหาได้จากสมการในรูป

$$x_i = \frac{c_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}}$$

Gauss Elimination Alg

- Back-Substitution

$$x_i = \frac{c_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}}$$

Backward-Substitution Pseudo Code:

```
x(n) = c(n)/a(n,n)
DOFOR i = n-1 to 1 step -1
    sum = 0
    DOFOR j = i+1 to n
        sum = sum + a(i,j)*x(j)
    ENDDO
    x(i) = (c(i)-sum)/a(i,i)
ENDDO
```

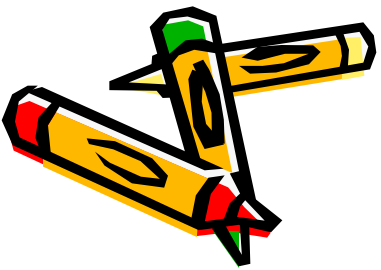
Example 8.1

Example 8.1 จงใช้ Gauss Elimination เพื่อแก้สมการ

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

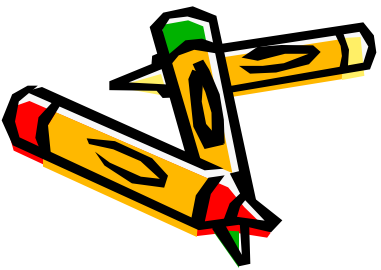
ต่อไปนี้เป็นผลลัพธ์จากการ Run Program

```
factor = 0.0333
a =
    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0   -0.1900   10.0200
c =
    7.8500
  -19.5617
   70.6150
```



Example 8.1

```
factor = 0.0333
a =
    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0   -0.1900   10.0200
c =
    7.8500
  -19.5617
   70.6150


factor =  -0.0271
a =
    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0         0   10.0120
c =
    7.8500
  -19.5617
   70.0843
```



Example 8.1

- Back Substitution

```
x =
     0
     0
  7.0000
x =
     0
 -2.5000
  7.0000
x =
  3.0000
 -2.5000
  7.0000
```

ปัญหาของ Gauss Elimination

ปัญหาเรื่อง Divide By Zero:

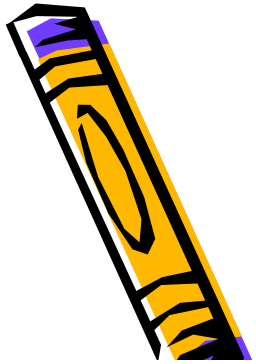
ในการกำจัด Unknown ออกนั้น เราต้องหาค่า Factor ที่มีการหารด้วย a_{ii} และในกรณีที่สมการดั้งเดิม ค่า a_{ii} มีค่าเป็นศูนย์ เราจะเจอปัญหาของ Divide by Zero ยกตัวอย่างเช่นในกรณีของ

$$2x_2 + 3x_3 = 8$$
$$4x_1 + 6x_2 + 7x_3 = -3$$
$$2x_1 + x_2 + 6x_3 = 5$$

แม้ว่าบางครั้งเราไม่เกิดปัญหา Divide by Zero แต่ถ้าค่า Coefficient เหล่านี้มีค่าใกล้ศูนย์มาก จะมีผลต่อ Accuracy ของคำตอบ วิธีการแก้ไขคือวิธีการที่เรียก Pivoting(รายละเอียดจะไม่กล่าวถึง) โดยสลับแถวของสมการ เช่น เปลี่ยนสมการในรูป

$$4x_1 + 6x_2 + 7x_3 = -3$$
$$2x_2 + 3x_3 = 8$$
$$2x_1 + x_2 + 6x_3 = 5$$

ปัญหาของ Gauss Elimination



ปัญหา Round-Off Error:

เนื่องจากการคำนวณเป็น Iteration และจะเกิด Error จากการปัดเศษของเลขหลักสุดท้ายในแต่ละขั้นตอน และลักษณะการทำงานของ Algorithm จะทำให้เกิดการสะสมของ Error เกิดขึ้น โดยเฉพาะสมการที่มีขนาดใหญ่ วิธีการแก้ไขคือการเพิ่ม Significant Digit ในการคำนวณ แต่วิธีที่ดีกว่าคือใช้คณิตศาสตร์ในการคำนวณที่ไม่ต้องมีการปัดเศษ คือการคำนวณแบบใช้เศษส่วน อย่างไรก็ตาม รายละเอียดเราจะไม่กล่าวถึง

ปัญหาของ Gauss Elimination

Ill-Conditioned system เกิดจากเมื่อมีการเปลี่ยนแปลงเล็กน้อยในค่า Coefficient ของระบบ จะทำให้ Solution ของระบบเปลี่ยนแปลงไปมาก ดังนั้นเมื่อมี Round-Off Error เพียงเล็กน้อย จะมีผลให้ Solution เปลี่ยนแปลงไป เนื่องจาก Round-Off Error ที่เกิด จะเปลี่ยนแปลงค่าที่แท้จริงของ Coefficient ที่คำนวณได้ ยกตัวอย่างสมการ

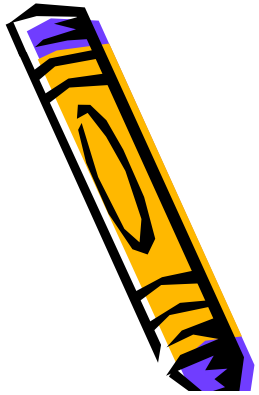
$$x_1 + 2x_2 = 10$$
$$1.1x_1 + 2x_2 = 10.4$$

ซึ่งมี Solution $x_1 = 4, x_2 = 3$ ถ้า Coefficient ของ x_1 ในสมการที่สองเปลี่ยนเพียงเล็กน้อย จาก 1.1 เป็น 1.05 ในสมการ

$$x_1 + 2x_2 = 10$$
$$1.05x_1 + 2x_2 = 10.4$$

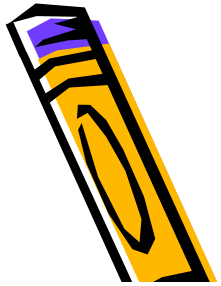
เราจะได้ Solution ที่เปลี่ยนแปลงไปมากคือ $x_1 = 8, x_2 = 1$ สังเกตว่า ค่าที่เปลี่ยนไปของ Coefficient มีเพียงเล็กน้อยเท่านั้น ซึ่งอาจจะเกิดจาก Round-Off Error หรือการเก็บข้อมูลที่ไม่แน่นอน

ปัญหาของ Gauss Elimination



เป็นการยากที่เราจะตรวจสอบว่าระบบเป็น Ill-Conditioned หรือไม่ สิ่งหนึ่งที่ดูได้คือการหาค่า Determinant ของ Matrix **A** ที่มีค่าใกล้เคียงศูนย์ แต่ว่ามันควรจะใกล้เคียงศูนย์แค่ไหนถึงจะเป็น Ill-Conditioned นั้นยากที่จะบอกได้ เพราะเราสามารถคูณสมการทั้งหมดด้วยค่า Constant ซึ่งค่า Determinant ก็จะมีค่าเป็นจำนวนเท่าตาม Factor ที่มาคูณด้วย แต่สมการที่ได้ยังคงเป็นสมการเดิม และมีคำตอบเดิม

Gauss-Jordan Method

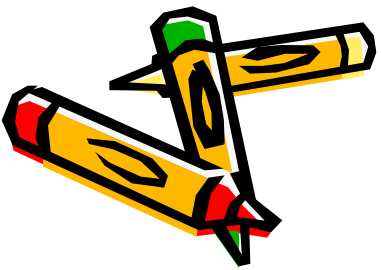


Gauss-Jordan Method เป็นวิธีที่ปรับมาจากวิธีของ Gauss Elimination ซึ่งจะทำการกำจัด Unknown ให้เหลือเพียงตัวเดียวในแต่ละสมการ ทำให้เราได้คำตอบทันที ดังนั้นในขบวนการ Elimination เราจะเหลือ Identity Matrix และไม่ต้องมี Back-Substitution เช่นจาก

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

เมื่อใช้ Gauss-Jordan Method เราจะได้

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1^n \\ c_2^n \\ c_3^n \end{bmatrix} \Rightarrow x_1 = c_1^n,\ x_2 = c_2^n,\ x_3 = c_3^n$$



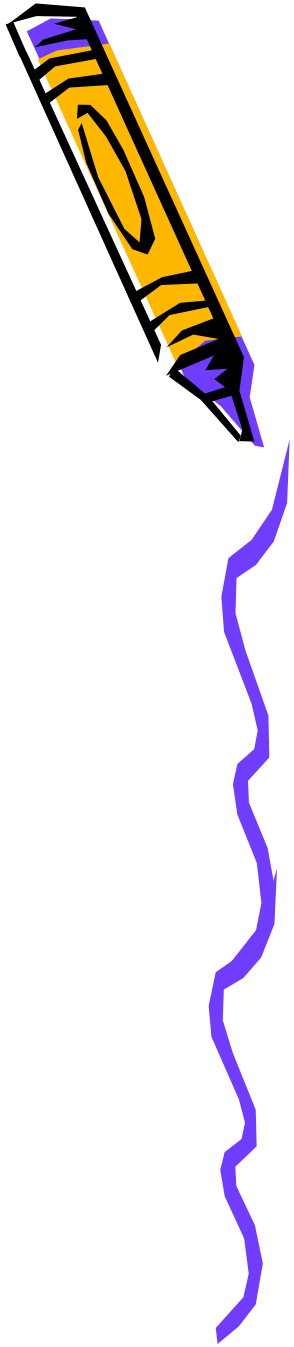
Gauss-Jordan Method

วิธีการก็คือ ในแต่ละ Step เมื่อเรากำจัด Unknown ออกแล้ว เราทำการ Scale ค่า a_{ii} ให้เท่ากับ 1 ในสมการที่ i จากนั้นจะนำสมการที่ $i+1$ มาทำการ Scale และไปลบออกจากสมการที่ i เพื่อกำจัด Unknown ในสมการ i ออกด้วย (ดูตัวอย่าง 7.2)

Algorithm สามารถเขียนเป็น Pseudo Code ได้ดังนี้(สมมุติ Matrix มีขนาด $n \times n+1$ โดย Column สุดท้ายคือ $\mathbf{C}$)

```
DOFOR k = 1 to n
   dummy = a(k,k)
   DOFOR j = 1 to n+1
      a(k,j) = a(k,j)/dummy
   ENDDO
   DOFOR i = 1 to n
      IF (i <> k)
         dummy = a(i,k)
         DOFOR j = 1 to n+1
            a(i,j) = a(i,j) - dummy*a(k,j)
         ENDDO
      ENDIF
   ENDDO
ENDDO
```

Example 8.2



Example 8.2 จงใช้ Gauss-Jordan เพื่อแก้สมการ

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

Example 8.2

$$\left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{array}\right]$$

$$(R2)-(R1)\times .1/3 \rightarrow (R2)$$

$$\left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0 & 7.0033 & -0.2933 & -19.5617 \\ 0.3 & -0.2 & 10 & 71.4 \end{array}\right]$$

$$(R3)-(R1)\times .3/3 \rightarrow (R3)$$

$$\left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0 & 7.0033 & -0.2933 & -19.5617 \\ 0 & -0.1900 & 10.0200 & 70.6150 \end{array}\right]$$

$$(R1)=(R1)/3$$

$$\left[\begin{array}{ccc|c} 1 & -0.0333 & -0.0667 & 2.6167 \\ 0 & 7.0033 & -0.2933 & -19.5617 \\ 0 & -0.1900 & 10.0200 & 70.6150 \end{array}\right]$$

Example 8.2

$$\left[\begin{array}{cccc} 1.0000 & -0.0333 & -0.0667 & 2.6167 \\ 0 & 7.0033 & -0.2933 & -19.5617 \\ 0 & -0.1900 & 10.0200 & 70.6150 \end{array}\right]$$

$$(R3) = (R3) - (R2) \times (-.19) / 7.0033$$

$$\left[\begin{array}{cccc} 1.0000 & -0.0333 & -0.0667 & 2.6167 \\ 0 & 7.0033 & -0.2933 & -19.5617 \\ 0 & 0 & 10.0120 & 70.0843 \end{array}\right]$$

$$(R2) = (R2) / 7.0033$$

$$\left[\begin{array}{cccc} 1.0000 & -0.0333 & -0.0667 & 2.6167 \\ 0 & 1 & -0.0419 & -2.7932 \\ 0 & 0 & 10.0120 & 70.0843 \end{array}\right]$$

$$(R1) = (R1) - (R2) \times (-.0333)$$

$$\left[\begin{array}{cccc} 1.0000 & 0 & -0.0681 & 2.5236 \\ 0 & 1.0000 & -0.0419 & -2.7932 \\ 0 & 0 & 10.0120 & 70.0843 \end{array}\right]$$

Example 8.2

$$\left[\begin{array}{ccc|c} 1.0000 & 0 & -0.0681 & 2.5236 \\ 0 & 1.0000 & -0.0419 & -2.7932 \\ 0 & 0 & 10.0120 & 70.0843 \end{array}\right]$$

$$(R3) = (R3)/10.012$$

$$\left[\begin{array}{ccc|c} 1.0000 & 0 & -0.0681 & 2.5236 \\ 0 & 1.0000 & -0.0419 & -2.7932 \\ 0 & 0 & 1.0000 & 7.0000 \end{array}\right]$$

$$(R2) = (R2) - (R3) * (-.0419)$$

$$\left[\begin{array}{ccc|c} 1.0000 & 0 & -0.0681 & 2.5236 \\ 0 & 1.0000 & 0 & -2.5000 \\ 0 & 0 & 1.0000 & 7.0000 \end{array}\right]$$

$$(R1) = (R1) - (R3) * (-.0681)$$

$$\left[\begin{array}{ccc|c} 1.0000 & 0 & 0 & 3.0000 \\ 0 & 1.0000 & 0 & -2.5000 \\ 0 & 0 & 1.0000 & 7.0000 \end{array}\right]$$

Example 8.2

$$\left[\begin{array}{ccc|c} 1.0000 & 0 & 0 & 3.0000 \\ 0 & 1.0000 & 0 & -2.5000 \\ 0 & 0 & 1.0000 & 7.0000 \end{array}\right]$$

$$\mathbf{A'X = C'}$$

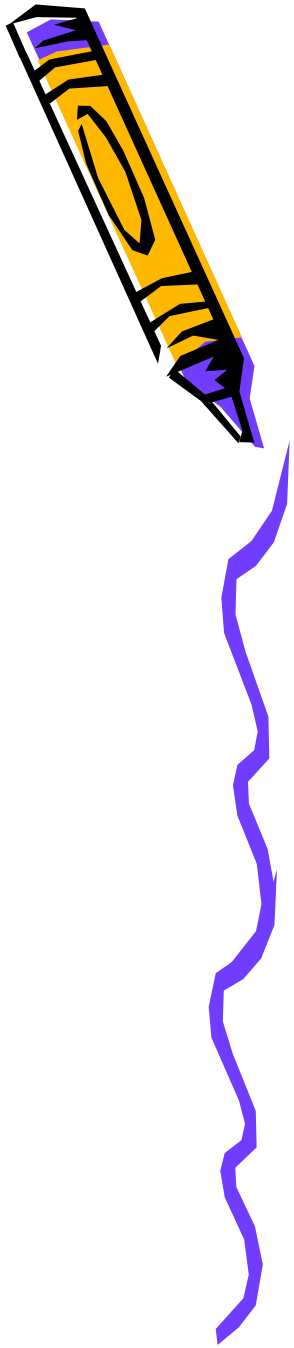
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2.5 \\ 7 \end{bmatrix}$$

$$x_1 = 3$$

$$x_2 = -2.5$$

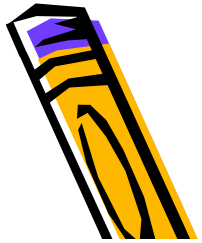
$$x_3 = 7$$

Example 8.2



```
k = 0
a | c =

    3.0000   -0.1000   -0.2000  |   7.8500
    0.1000    7.0000   -0.3000  | -19.3000
    0.3000   -0.2000   10.0000  |  71.4000
k =1
a | c =

    1.0000   -0.0333   -0.0667  |   2.6167
         0    7.0033   -0.2933  | -19.5617
         0   -0.1900   10.0200  |  70.6150

k = 2
a | c =

    1.0000         0   -0.0681  |  2.5236
         0    1.0000   -0.0419  | -2.7932
         0         0   10.0120  | 70.0843

k = 3
a | c =

    1.0000         0         0  |  3.0000
         0    1.0000         0  | -2.5000
         0         0    1.0000  |  7.0000
```

การหา Matrix Inverse ด้วย GJ



ปกติแล้วการหา Inverse ของ Matrix คือการหา Solution ของสมการ $\mathbf{AX} = \mathbf{I}$ เช่นในกรณีของ 3×3 Matrix เราได้

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ซึ่งที่จริงแล้วเราได้ $\mathbf{X} = \mathbf{A}^{-1}$ นั่นเอง และในกรณีนี้ Unknown $\mathbf{X}$ เป็น Matrix ไม่ใช่ Vector

ถ้าพิจารณาให้ดีจากสมการข้างบน โดยใช้หลักการคูณกันของ Matrix การหา Inverse ก็คือการแก้สมการ Linear n สมการพร้อมๆกัน ในตัวอย่างข้างบนเราแก้สมการ 3 สมการดังนี้

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

ซึ่งคำตอบของสมการนี้ เราจะได้ Column แรกของ Inverse ของ Matrix จากนั้นเราหา Solution ของ

การหา Matrix Inverse ด้วย GJ

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{22} \\ x_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

และคำตอบจะได้ Column ที่สองของ Inverse ของ Matrix สุดท้ายเราแก้สมการ

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

เราจะได้ Column สุดท้ายของ Matrix Inverse

การหา Matrix Inverse ด้วย GJ

จากหลักการที่อธิบายข้างบน เราสามารถใช้ Algorithm ใดๆก็ได้ที่ใช้ในการแก้สมการ Linear Equation มาดัดแปลงหา Inverse ของ Matrix อย่างไรก็ตาม เราจะต้องกระทำถึง n ครั้ง เพื่อจะได้คำตอบ และจะใช้การคำนวณมากสำหรับการหา Inverse ของ Matrix ขนาดใหญ่

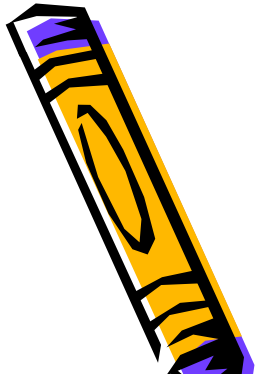
ถ้าเรามาดูวิธีของ Gauss-Elimination เมื่อนำมาหา Matrix Inverse จะพบว่าส่วนของ Forward Substitution สามารถกระทำไปพร้อมกันได้หมด โดยเราตั้งสมการดังนี้(ใช้ตัวอย่างเดิม)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ซึ่ง Step ของ Forward Elimination สามารถกระทำทีเดียวพร้อมๆกัน และเราจะได้

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ c'_{21} & c'_{22} & c'_{23} \\ c''_{31} & c''_{32} & c''_{33} \end{bmatrix}$$

การหา Matrix Inverse ด้วย GJ



เทียบสัดส่วนของการคำนวณแล้ว ส่วนของ Forward Elimination จะใช้คิดเป็นอัตราส่วนมากกว่า 90 % แต่เมื่อนำมาใช้ในการหา Inverse แล้ว จำนวนของ Operation จะมีมากเพิ่มขึ้นในส่วนของ Back-Substitution เนื่องจากเราต้องทำถึง n ครั้ง ทำให้ไม่เหมาะสมสำหรับ Matrix ขนาดใหญ่

ในกรรมวิธีของ Gauss-Jordan นั้น เนื่องจากไม่มีการทำ Back-Substitution ทำให้การดัดแปลง Algorithm เมื่อมาใช้หา Inverse ของ Matrix สามารถกระทำไปพร้อมๆกันภายในขั้นตอนเดียว ซึ่งเราจะได้โปรแกรมที่รวบรัด และรวดเร็ว ซึ่งเป็นวิธีที่นิยมมากอันหนึ่งในการหา Inverse ของ Matrix

ในการนำกรรมวิธีของ Gauss-Jordan มาใช้หา Inverse ของ Matrix นั้น เราเขียน $\mathbf{C}$ ให้เป็น Identity Matrix แทนที่จะเป็น Column Vector ในรูปของ

การหา Matrix Inverse ด้วย GJ

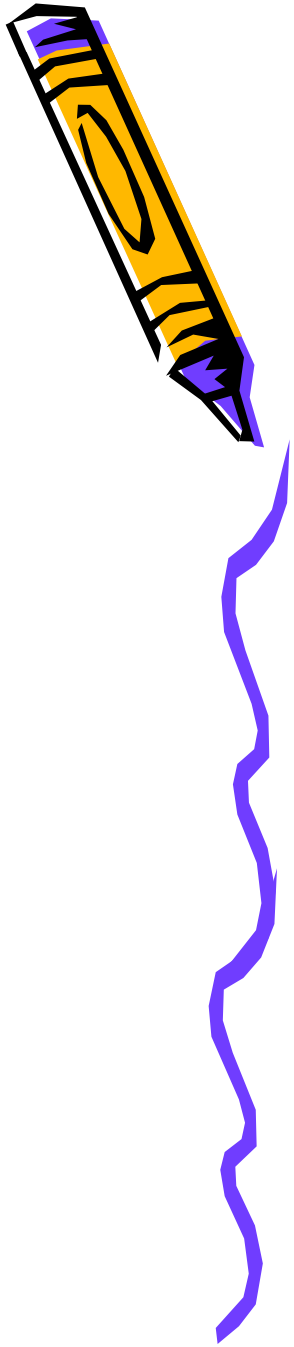
$$[\mathbf{A}:\mathbf{C}]=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

จากนั้นเมื่อเราใช้ Gauss-Jordan เพื่อลดรูป Matrix **A** ให้อยู่ในรูป Identity Matrix พร้อมๆกันนั้น เราจะทำการเปลี่ยน Identity Matrix เดิมให้อยู่ในรูปของ Inverse ของ Matrix **A** กล่าวคือ

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11}^{-1} & a_{12}^{-1} & a_{13}^{-1} \\ a_{21}^{-1} & a_{22}^{-1} & a_{23}^{-1} \\ a_{31}^{-1} & a_{32}^{-1} & a_{33}^{-1} \end{bmatrix}$$

ซึ่งจะเห็นได้ว่าวิธีของ Gauss-Jordan มีความรวบรัดมากกว่า

Example 8.3



Example 8.3 จงใช้ Gauss-Jordan เพื่อหา Inverse ของ

$$\mathbf{A} = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

Example 8.3

```
k = 0
    3.0000   -0.1000   -0.2000    1.0000         0         0
    0.1000    7.0000   -0.3000         0    1.0000         0
    0.3000   -0.2000   10.0000         0         0    1.0000
k = 1
ans =

    1.0000   -0.0333   -0.0667    0.3333         0         0
         0    7.0033   -0.2933   -0.0333    1.0000         0
         0   -0.1900   10.0200   -0.1000         0    1.0000
k = 2
ans =
    1.0000         0   -0.0681    0.3332    0.0048         0
         0    1.0000   -0.0419   -0.0048    0.1428         0
         0         0   10.0120   -0.1009    0.0271    1.0000
k =  3
ans =
    1.0000         0         0    0.3325    0.0049    0.0068
         0    1.0000         0   -0.0052    0.1429    0.0042
         0         0    1.0000   -0.0101    0.0027    0.0999
```

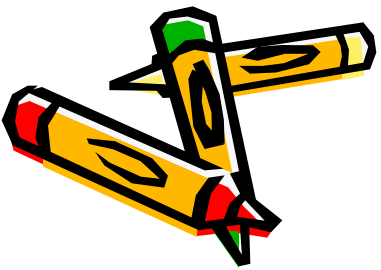
Example 8.3

```
ans =
    1.0000         0         0    0.3325    0.0049    0.0068
         0    1.0000         0   -0.0052    0.1429    0.0042
         0         0    1.0000   -0.0101    0.0027    0.0999
```

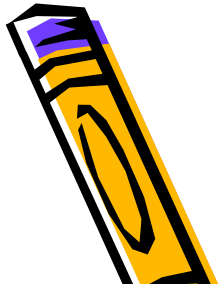
$$\begin{bmatrix} 3 & -.1 & -.2 \\ .1 & 7 & -.3 \\ .3 & -.2 & 10 \end{bmatrix}^{-1} = \begin{bmatrix} .3325 & .0049 & .0068 \\ -.0052 & .1429 & .0042 \\ -.0101 & .0027 & .0999 \end{bmatrix}$$

จำนวนการคำนวณจะใช้น้อยกว่าวิธีทาง Analytic Method มาก

$$\mathbf{A}^{-1} = \frac{Adj\mathbf{A}}{|\mathbf{A}|}$$



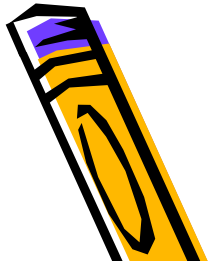
Iterative Method and Gauss-Seidel



Gauss-Jordan สามารถใช้หา Inverse ของ Matrix หรือแก้สมการที่มีถึง 100 Unknown ถ้าเราตั้ง Significant Digit ให้เหมาะสม และถ้าระบบไม่เป็น Ill-Conditioned อย่างไรก็ตาม ในระบบที่มีขนาดใหญ่กว่านี้ จะเกิดปัญหาของ Round-Off Error และวิธีการแบบ Elimination จะไม่เหมาะสม เราจำเป็นจะต้องพึ่งวิธีทาง Iterative หรือวิธี Approximate แทน เนื่องจากวิธีทาง Iterative นั้นสามารถกระทำต่อได้เรื่อยๆ และจะหยุดเมื่อได้ Estimate Error น้อยกว่าค่า e_s ที่กำหนด(ดูบทที่ 7) ผิดกับวิธีของ Elimination ที่จะมี Loop ของการคำนวณที่แน่นอน ตามขนาดของระบบ อย่างไรก็ตามพึงเข้าใจก่อนว่าวิธีทาง Iterative นั้นจัดว่าเป็นวิธี Approximate เพราะเราจะไม่ได้คำตอบที่แท้จริง เพียงแต่ว่าคำตอบจะเข้าใกล้ค่าที่แท้จริงเรื่อยๆ ถ้าระบบ Converge และในทางทฤษฎีแล้ว เราจะต้อง Run ถึง Infinity Iteration ถึงจะได้คำตอบที่แท้จริง

วิธีของ Gauss-Seidel เป็นวิธีทาง Iterative สำหรับการแก้ปัญหาของระบบของสมการที่นิยมใช้กันมากที่สุด โดยทำการหา Solution จากระบบสมการเดิมด้วยการหาค่า x_1 ในสมการที่หนึ่ง ค่า x_2 ในสมการที่สอง จนถึง x_n ในสมการสุดท้าย จากนั้นจึงทำ Iteration ซึ่งความจริงก็คือวิธีของ Simple One-Point Iteration สำหรับหลาย Unknown และมีรายละเอียดต่อไปนี้

Gauss-Seidel



จากสมการ ของระบบ Linear Algebraic Equation ที่มี n Unknown $x_1, x_2, \ldots, x_n$ สามารถเขียนในรูป Matrix

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = c_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = c_2 \\
&\quad\vdots \qquad\quad \vdots \qquad\quad \vdots \\
&a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = c_n
\end{aligned}
\quad = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \mathbf{AX} = \mathbf{C}
$$

Gauss-Seidel

เมื่อเราแก้สมการหาค่า x_i จากแถวที่ i เราได้

$$x_1 = \frac{c_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n}{a_{11}}$$

$$x_2 = \frac{c_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n}{a_{22}}$$

$$x_3 = \frac{c_3 - a_{31}x_1 - a_{32}x_2 - \cdots - a_{3n}x_n}{a_{33}}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$x_n = \frac{c_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{nn-1}x_{n-1}}{a_{nn}}$$

Gauss-Seidel

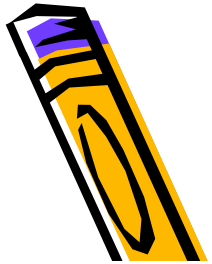
จากนั้นเราเริ่มขบวนการโดยเดาค่า x_i เริ่มต้น ทั้งหมด n ค่า ซึ่งปกติจะเริ่มจากให้ x_i เริ่มต้นเป็นศูนย์ทั้งหมด และใน Iteration แรก เราคำนวณค่าใหม่ของ $x_1^1, x_2^1, \cdots, x_n^1$ โดยตัวเลข Superscript หมายถึงค่า x_i^j ที่ Estimate ใหม่ใน Iteration ที่ j และโปรแกรมจะ Converge ถ้าค่า Estimate Error สำหรับทุกๆ Unknown มีค่าลดลง โปรแกรมจะหยุดเมื่อเราได้ Estimate Error ทุกตัวน้อยกว่าค่า e_s ที่ตั้งไว้

ค่า Estimate Error สำหรับ Unknown x_i^j ที่ Iteration j สามารถคำนวณได้จาก

$$\left|e_{a,j}\right| = \left|\frac{x_i^j - x_i^{j-1}}{x_i^j}\right| \times 100\%$$

เนื่องจาก Gauss-Seidel Algorithm อาจจะไม่ Converge หรือ Converge ช้าเราสามารถปรับปรุงวิธีการให้ Converge เร็วขึ้น โดยแทนที่จะใช้ค่าใหม่ที่ Estimate ได้โดยตรง แต่เราใช้ Weight Sum ของค่าใหม่กับค่าเดิม ซึ่งเราเรียกว่าเป็นการทำ Relaxation ดังนี้

Gauss-Seidel



$$x_i^{New} = \lambda x_i^{New} + (1-\lambda)x_i^{Old}$$

โดยค่า λ มีค่าอยู่ระหว่าง ศูนย์และสอง ถ้า $\lambda = 1$ เราจะได้วิธีปกติของ Gauss-Seidel ที่อธิบายมาแล้ว แต่ถ้า λ มีค่าอยู่ระหว่าง 0 และ 1 เราเรียกว่าเป็น ***Underrelaxation*** และจะใช้ในกรณีที่จะทำให้ระบบที่ไม่ Converge เป็นระบบที่สามารถ Converge หรือ ในกรณีที่คำตอบ Converge ช้าเนื่องจากมีการ Oscillation ของคำตอบกลับไปกลับมา และถ้าค่า λ มีค่าอยู่ระหว่าง 1 ถึง 2 เราเรียกว่าเป็น Overrelaxation เพราะเป็นการเพิ่มค่าน้ำหนักให้กับค่าใหม่ที่มากกว่าเดิม กรณีนี้จะใช้กับระบบที่มีการ Converge อยู่แล้ว แต่จะช่วยให้มัน Converge ได้เร็วยิ่งขึ้น

Algorithm ของ Gauss Seidel เขียนเป็น Pseudo Code ได้ดังนี้ (รวมถึงการ Relaxation ด้วย)

```
DOFOR i = 1 to n
   dummy = a(i,i)
   DOFOR j = 1 to n
      a(i,j) = a(i,j)/dummy
   ENDDO
   c(i) = c(i)/dummy
ENDDO
sentinel = 0
iter = 0
DOWHILE (iter < maxit) and (sentinel = 0)
   sentinel = 1
   iter = iter + 1
   DOFOR i = 1 to n
      old = x(i)
      sum = c(i)
      DOFOR j = 1 to n
         IF i <> j
            sum = sum - a(i,j)*x(j)
         ENDIF
      ENDDO
      x(i) = Lambda*sum + (1-Lambda)*old
      IF (sentinel = 1)and (x(i) <> 0)
         ea = abs((x(i)-old)/x(i))*100
         IF ea > es
            sentinel = 0
         ENDIF
      ENDIF
   ENDDO
ENDDO
```

ในภาคผนวกของบทนี้แสดงตัวอย่างของ MATLAB Program ที่มีการ Relaxation

Gauss-Seidel: Ex 8.4

Example 8.4 จงใช้ Gauss-Seidel เพื่อแก้สมการ

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

เมื่อเรียงสมการใหม่เพื่อหาค่า Unknown เราได้

$$x_1 = \frac{7.85 + 0.1x_2 + 0.2x_3}{3}$$

$$x_2 = \frac{-19.3 - 0.1x_1 + 0.3x_3}{7}$$

$$x_3 = \frac{71.4 - 0.3x_1 + 0.2x_2}{10}$$

Iteration ที่ 1

Gauss-Seidel: Ex 8.4

เริ่มจาก $x_2 = 0, x_3 = 0$ เราคำนวณหา x_1 จากสมการแรก เราได้ $x_1 = 7.85/3 = 2.61667$

จากค่า $x_1 = 2.61667, x_3 = 0$ เราคำนวณ x_2 จากสมการที่ 2 เราได้ $x_2 = -2.79452$

และจาก $x_1 = 2.61667, x_2 = -2.79452$ เราคำนวณค่า x_3 จากสมการสุดท้าย ได้ $x_3 = 7.00561$

Iteration ที่ 2, 3, …

คำนวณเหมือนเดิม แต่ใช้ค่า x_1, x_2, x_3 ที่ได้ใหม่ล่าสุด แทนค่าลงในสมการที่หนึ่ง สอง และ สาม

ข้างล่างเป็นผลจากการ Run Program โดยตั้ง $e_s = 0.01\%$ และไม่มี Relaxation (โดยการตั้งค่า $\lambda = 1$)

Gauss-Seidel: Ex 8.4

$$x_1^0 = x_2^0 = x_3^0 = 0$$

$$x_1^1 = \frac{7.85 + 0.1x_2^0 + 0.2x_3^0}{3} = \frac{7.85 + 0.1(0) + 0.2(0)}{3} = \frac{7.85}{3} = 2.61667$$

$$x_2^1 = \frac{-19.3 - 0.1x_1^1 + 0.3x_3^0}{7} = \frac{-19.3 - 0.1(2.61667) + 0.3(0)}{7} = \frac{-19.56167}{7} = -2.79452$$

$$x_3^1 = \frac{71.4 - 0.3x_1^1 + 0.2x_2^1}{10} = \frac{71.4 - 0.3(2.61667) + 0.2(-2.79452)}{10} = \frac{70.056095}{10} = 7.00561$$

$$e_{a,1} = \left| \frac{2.61667 - 0}{2.61667} \times 100 \right| = 100\%$$

$$e_{a,1} = \left| \frac{-2.79452 - 0}{-2.79452} \times 100 \right| = 100\%$$

$$e_{a,1} = \left| \frac{7.00561 - 0}{7.00561} \times 100 \right| = 100\%$$

Gauss-Seidel: Ex 8.4

$$x_1^1 = 2.61667,\ x_2^1 = -2.79452,\ x_3^1 = 7.00561$$

$$x_1^2 = \frac{7.85 + 0.1x_2^1 + 0.2x_3^1}{3} = \frac{7.85 + 0.1(-2.79452) + 0.2(7.00561)}{3} = \frac{8.9716968}{3} = 2.99056$$

$$x_2^2 = \frac{-19.3 - 0.1x_1^2 + 0.3x_3^1}{7} = \frac{-19.3 - 0.1(2.99056) + 0.3(7.00561)}{7} = \frac{-17.49737}{7} = -2.49962$$

$$x_3^2 = \frac{71.4 - 0.3x_1^2 + 0.2x_2^2}{10} = \frac{71.4 - 0.3(2.99056) + 0.2(-2.49962)}{10} = \frac{70.00291}{10} = 7.00029$$

$$e_{a,2} = \left| \frac{2.99056 - 2.61667}{2.99056} \times 100 \right| = 12.50\%$$

$$e_{a,2} = \left| \frac{-2.49962 - (-2.79452)}{-2.49962} \times 100 \right| = 11.80\%$$

$$e_{a,2} = \left| \frac{7.00029 - 7.00561}{7.00029} \times 100 \right| = 0.076\%$$

Gauss-Seidel: Ex 8.4

$$x_1^2 = 2.99056,\ x_2^2 = -2.49962,\ x_3^2 = 7.00029$$

$$x_1^3 = \frac{7.85 + 0.1x_2^2 + 0.2x_3^2}{3} = \frac{7.85 + 0.1(-2.49962) + 0.2(7.00029)}{3} = \frac{9.0001}{3} = 3.00003$$

$$x_2^3 = \frac{-19.3 - 0.1x_1^3 + 0.3x_3^2}{7} = \frac{-19.3 - 0.1(3.00003) + 0.3(7.00029)}{7} = \frac{-17.499916}{7} = -2.499988$$

$$x_3^3 = \frac{71.4 - 0.3x_1^3 + 0.2x_2^3}{10} = \frac{71.4 - 0.3(3.00003) + 0.2(-2.499988)}{10} = \frac{69.99999}{10} = 6.999999$$

$$e_{a,3} = \left| \frac{3.00003 - 2.99056}{3.00003} \times 100 \right| = 0.3157\%$$

$$e_{a,3} = \left| \frac{-2.499988 - (-2.49962)}{-2.499988} \times 100 \right| = 0.01472\%$$

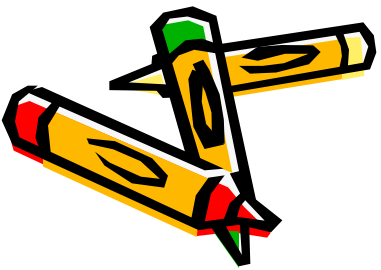
$$e_{a,3} = \left| \frac{6.999999 - 7.00029}{6.999999} \times 100 \right| = 0.004157\%$$

Gauss-Seidel: Ex 8.4

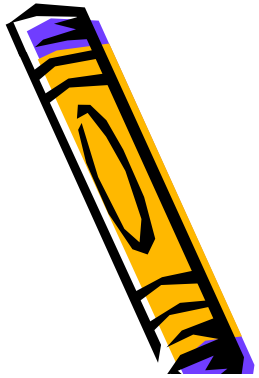
```
iter =  0

x =
      0
      0
      0

iter =  1
x =
    2.6167
   -2.7945
    7.0056
ea =
   100
   100
   100
iter =  2
x =
    2.9906
   -2.4996
    7.0003
ea =
   12.5023
   11.7977
    0.0760
```

Gauss-Seidel: Ex 8.4

```
iter =  3
x =
    3.0000
   -2.5000
    7.0000
ea =
    0.3158
    0.0145
    0.0042
iter =  4
x =
    3.0000
   -2.5000
    7.0000
ea =
    0.0011
    0.0005
    0.0000
```



Jacobi Method



เราจะจบวิธีของ Iterative Method ด้วยการกล่าวถึง Variation ของ Gauss-Seidel ที่ชื่อ Jacobi Method ซึ่งในวิธีของ Gauss-Seidel ที่กล่าวมานั้น เราใช้ค่า x_i^j และ x_i^{j-1} ที่คำนวณได้ล่าสุดในการหา x_{i+1}^j ดังนั้นถ้าระบบมีการ Converge เราจะได้ค่า Estimate ที่ดีที่สุดของ Unknown ตัวต่อไป และจะทำให้เกิดการ Converge ได้อย่างรวดเร็ว อย่างไรก็ตามถ้าเราใช้ชุดของ x_i^j ทั้งหมดมาทำการ Estimate ค่าชุดใหม่ของ x_i^{j+1} เราจะได้กรรมวิธีที่ชื่อ Jacobi Method

ที่จริงแล้ว Gauss-Seidel Method เป็นวิธีที่ปรับปรุงมาจาก Jacobi Method และปกติจะให้การ Converge ที่รวดเร็วกว่าและปกติเป็นวิธีที่นิยมใช้กัน แต่ก็มีบางกรณีที่วิธีของ Jacobi กลับให้ผลที่รวดเร็วกว่า

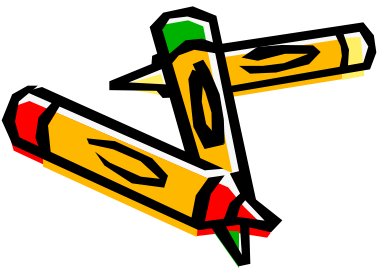
Convergence of Iterative Method

เมื่อพูดถึงการ Convergence ซึ่งเป็นสิ่งที่สำคัญสำหรับ Iterative Method เพราะวิธีนี้เป็นวิธีที่ปรับปรุงมาจากวิธีของ Simple One-Point Iteration สำหรับชุดของสมการ Linear ที่มี n Dimension เราสามารถพิสูจน์ได้ว่า Sufficient Condition ในการ Converge (แต่ไม่ใช่ Necessary Condition) นั้นคือถ้าระบบเป็น ***Diagonally Dominant*** กล่าวคือค่าของ Coefficient ที่จุด Pivot ทุกตัว(ค่า a_{ii}) ในระบบสมการมีค่ามากกว่าผลรวมของค่า Absolute ของ Coefficient อื่นในแถวเดียวกัน ดังนี้

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|$$

อย่างไรก็ตาม ในปัญหาทั่วไปแล้วปกติเป็นการยากที่จะสลับแถวของสมการหรือมีการบวกลบแถว เพื่อทำให้ระบบเป็น Diagonally Dominant รายละเอียดเราจะไม่กล่าวถึง

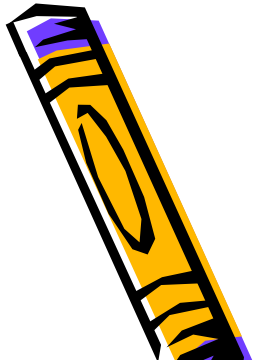
อีกวิธีหนึ่งที่จะทำให้ระบบ Converge ก็คือใช้การ Relaxation ซึ่งได้กล่าวมาแล้ว และการเลือกค่า λ ที่ถูกต้องถือว่าเป็นเรื่องสำคัญ รายละเอียดจะเกินเนื้อหาของวิชานี้



Break

- After Break
  - LU Decomposition

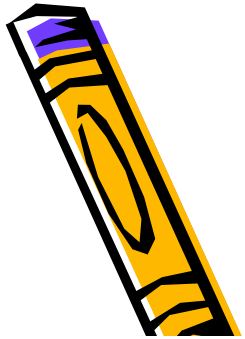
LU Decomposition



สรุปจากที่กล่าวมาแล้ว เรามีวิธีในการแก้ปัญหาระบบ Linear Algebraic Equation 2 แบบ คือวิธี Elimination Method (Gauss Elimination, Gauss-Jordan) และวิธี Iterative Method (Jacobi, Gauss-Seidel) ในส่วนนี้เราจะมากล่าวต่อในเรื่องของ Elimination Method ที่เรียกเทคนิคการทำ LU Decomposition

วิธีของ LU Decomposition จะมีประสิทธิภาพสูงกว่า เพราะจะปรับปรุงในส่วนของ Elimination Step และจะใช้ได้ดีกว่า Gauss-Jordan ในการหา Inverse ของ Matrix ทำให้วิธีนี้เป็นวิธีที่ใช้กันมากที่สุดในการแก้ปัญหาของระบบของ Linear Algebraic Equations

LU Decomposition



LU Decomposition จะเจอปัญหาเรื่อง Divide by Zero เช่นเดียวกับ Gauss Elimination และจะต้องมีการทำ Pivoting เพื่อป้องกัน อย่างไรก็ตามที่จะอธิบายต่อไปนี้จะถือว่า Coefficient Matrix มีค่า a_{ii} ทุกตัวไม่เท่ากับศูนย์

จากระบบของสมการ Linear เขียนในรูป Matrix ได้เป็น

$$\mathbf{AX} = \mathbf{C}$$

และสามารถจัดเรียงใหม่ได้เป็น

$$\mathbf{AX} - \mathbf{C} = \mathbf{0}$$

ถ้าสมมุติเราใช้ Gauss Elimination เปลี่ยน Coefficient Matrix ให้เป็น Upper Diagonal และทำการ Scale ให้ค่า Coefficient ในส่วน Diagonal มีค่าเท่ากับหนึ่ง ยกตัวอย่างในระบบที่มี 4 สมการ ดังนี้

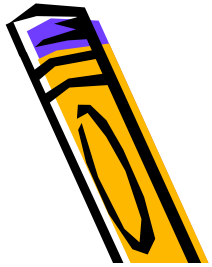
LU Decomposition

$$\begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

เมื่อเขียนในรูปแบบของ Matrix เราจะได้

$$\mathbf{UX - D = 0}$$

LU Decomposition



ครานี้สมมุติว่าเรามี Lower Diagonal Matrix $\mathbf{L}$ ดังนี้

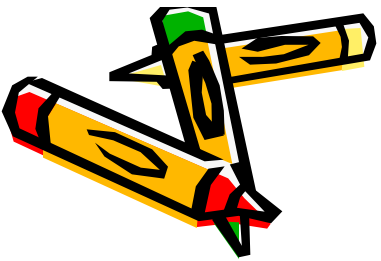
$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}$$

โดยที่ Lower Diagonal Matrix ดังกล่าวมีคุณสมบัติที่เมื่อคูณกับสมการข้างบนของ Upper Diagonal Matrix แล้วได้สมการเดิมกลับคืนมา นั่นคือ

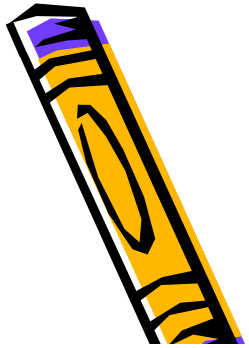
$$\mathbf{L}[\mathbf{UX} - \mathbf{D}] = \mathbf{AX} - \mathbf{C}$$

ซึ่งถ้าสมการเป็นจริง เราสามารถ Equate สองส่วนของสมการ จากทฤษฎีในการคูณกันของ Matrix ได้ดังนี้

$$\mathbf{LU} = \mathbf{A} \text{ และ } \mathbf{LD} = \mathbf{C}$$



LU Decomposition



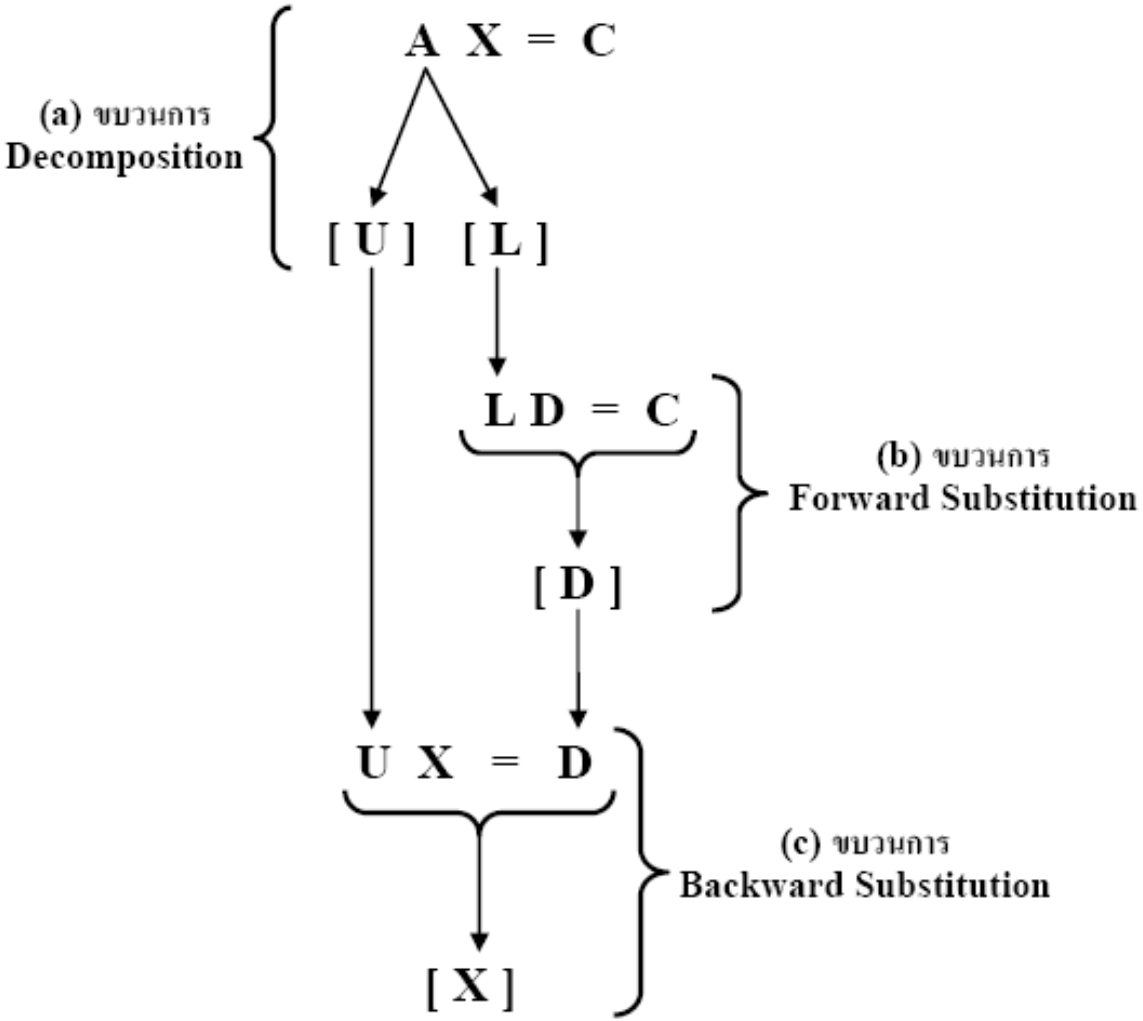
$$\mathbf{L[UX - D] = AX - C}$$

ซึ่งถ้าสมการเป็นจริง เราสามารถ Equate สองส่วนของสมการ จากทฤษฎีในการคูณกันของ Matrix ได้ดังนี้

$$\mathbf{LU = A} \text{ และ } \mathbf{LD = C}$$

สมการแรกข้างบนเรียกว่าเป็น LU Decomposition ของ Matrix $\mathbf{A}$ และถ้าเราสามารถหาได้ เราสามารถหา Solution ของสมการได้อย่างมีประสิทธิภาพ ด้วยขบวนการ 2 Step Substitution (Forward Substitution และ Backward Substitution) โดยใช้สมการ $\mathbf{UX - D = 0}$ และสมการ $\mathbf{LD = C}$ ดัง Diagram ในรูป

LU Decomposition



LU Decomposition

8.5.2 การดัดแปลง Gauss Elimination มาใช้ใน LU Decomposition

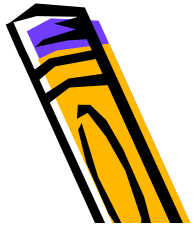
ในขบวนการ Gauss Elimination นั้น เราสร้าง Upper Diagonal Matrix ในรูป (ยกตัวอย่างกรณี 3×3 Matrix)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix}$$

จากสมการดั้งเดิมคือ

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

LU Decomposition



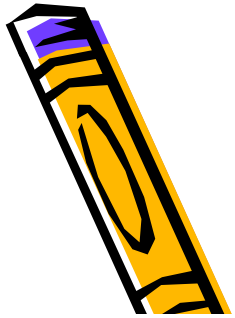
ซึ่งขั้นตอนแรกนั้นเราทำการคูณแถวที่ 1 ด้วย Factor $f_{21} = a_{21} / a_{11}$ จากนั้นนำผลที่ได้มาลบออกจากสมการในแถวที่สองเพื่อกำจัด a_{21} จากนั้นเราทำการกำจัด a_{31} ออก ด้วยการคูณแถวที่ 1 เช่นกันด้วย Factor $f_{31} = a_{31} / a_{11}$ และหักลบจากสมการในแถวที่ 3

ในรอบที่สอง เรากำจัด a'_{32} ออกด้วยการคูณสมการแถวที่สองที่ได้จากรอบแรกด้วย Factor $f_{32} = a'_{32} / a'_{22}$ ซึ่งที่จริงแล้วค่า Factor ที่ใช้ดังกล่าวคือค่า Coefficient ของ Lower Diagonal Matrix

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix}$$

ขอให้นักศึกษาลองทดสอบดูว่าความจริงแล้ว เราได้ $\mathbf{LU} = \mathbf{A}$ สังเกตอีกอย่างหนึ่งว่า Matrix ที่เราได้จาก Gauss Elimination จะต่างจากที่เราอธิบายในตอนต้นเล็กน้อย ซึ่งมีค่าในส่วนของ Diagonal เป็นหนึ่งใน $\mathbf{L}$ Matrix แทนที่จะเป็น $\mathbf{U}$ Matrix ซึ่งเราเรียกว่าเป็น ***Doolittle Decomposition*** หรือ ***Factorization*** ในขณะที่แบบที่อธิบายในตอนต้นเราเรียก ***Crout Decomposition***

LU Decomposition

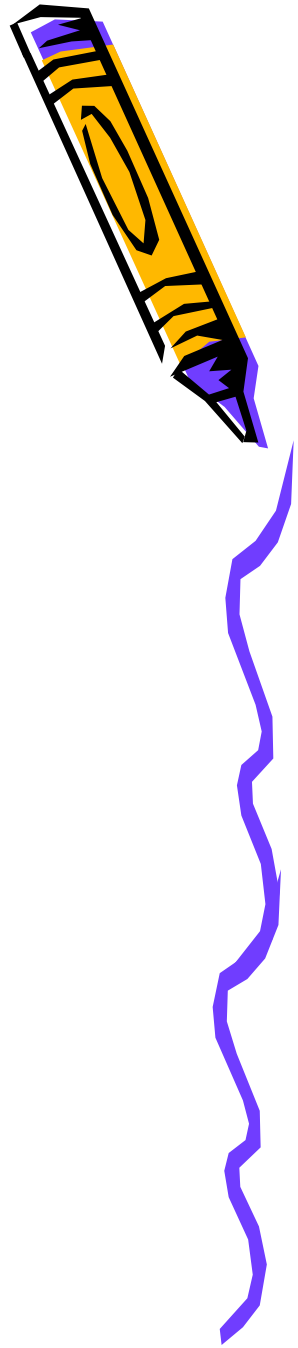


Example 8.5 จงใช้ Gauss Elimination เพื่อทำ LU Decomposition ของ Matrix

$$\mathbf{A} = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

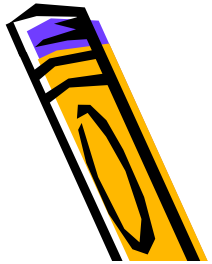
ต่อไปนี้เป็นผลที่ได้จากการ Run Program ที่แสดงในภาคผนวก โดยที่ 3 Column แรกคือ Matrix **L** และสาม Column ถัดมาคือ Matrix **U**

LU Decomposition



```
k = 1

ans =

    1.0000         0         0    3.0000   -0.1000   -0.2000
    0.0333    1.0000         0         0    7.0033   -0.2933
    0.1000         0    1.0000         0   -0.1900   10.0200

k = 2

ans =

    1.0000         0         0    3.0000   -0.1000   -0.2000
    0.0333    1.0000         0         0    7.0033   -0.2933
    0.1000   -0.0271    1.0000         0         0   10.0120
```

Crout Decomposition



8.5.3 Crout Decomposition

ในขบวนการของ Gauss Elimination นั้น การคำนวณส่วนใหญ่จะอยู่ที่ Forward Elimination โดยเฉพาะอย่างยิ่งถ้าปัญหามีขนาดใหญ่ ดังนั้นจึงมีความพยายามที่จะปรับปรุงขบวนการให้ดีขึ้น และแยกส่วนการคำนวณด้านซ้ายออกมา

วิธีที่มีประสิทธิภาพมากที่สุดอันหนึ่งก็คือ Crout Decomposition ซึ่งจะแยก Matrix ออกเป็น Upper Triagular และ Lower Triangular ยกตัวอย่างเช่น Matrix ขนาด 4×4

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Crout Decomposition

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Crout Decomposition จะเริ่มจากการคูณกันของ Matrix ด้านซ้ายของสมการ จากนั้น Equate ผลที่ได้ให้เท่ากับด้านขวาของสมการ ซึ่งขั้นแรกเราได้ว่า

$$l_{11} = a_{11}, \quad l_{21} = a_{21}, \quad l_{31} = a_{31}, \quad l_{41} = a_{41}$$

กล่าวคือ Column แรกของ $\mathbf{L}$ จะเท่ากับ Column แรกของ $\mathbf{A}$ หรือ $l_{i1} = a_{i1}, i = 1,2,\ldots,n$

Crout Decomposition

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

จากนั้นพิจารณาจากแถวแรกของ **L** ที่ต้องคูณกับแต่ละ Column ของ **U** และแก้สมการหาแถวแรกของ **U** ออกมา เราได้

$$l_{11} = a_{11}, \quad l_{11}u_{12} = a_{12}, \quad l_{11}u_{13} = a_{13}, \quad l_{11}u_{14} = a_{14}$$

และ

$$u_{12} = \frac{a_{12}}{l_{11}}, \quad u_{13} = \frac{a_{13}}{l_{11}}, \quad u_{14} = \frac{a_{14}}{l_{11}}$$

เราสรุปได้ว่า $u_{ij} = \dfrac{a_{1j}}{l_{11}}, \; j = 2,3,\ldots,n$

เมื่อถึงขั้นนี้ เราได้ Column แรกของ **L** และแถวแรกของ **U** จากนั้นขบวนการจะทำซ้ำสำหรับ Column ที่สอง และแถวที่สองของ **L** และ **U** ตามลำดับ ซึ่งเราจะได้

Crout Decomposition

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

เมื่อถึงขั้นนี้ เราได้ Column แรกของ **L** และแถวแรกของ **U** จากนั้นขบวนการจะทำซ้ำสำหรับ Column ที่สอง และแถวที่สองของ **L** และ **U** ตามลำดับ ซึ่งเราจะได้

$$l_{i2} = a_{i2} - l_{i1}u_{12}, \; i = 2,3,\ldots,n$$

$$u_{2j} = \frac{a_{2j} - l_{21}u_{1j}}{l_{22}}, \; j = 3,4,\ldots,n$$

นักศึกษาสามารถพิจารณาเองและหาคำตอบในขั้นต่อไปได้ดังนี้

$$l_{i3} = a_i - l_{i1}u_{13} - l_{i2}u_{23}, \;\; i = 3,4,\ldots,n$$

$$u_{3j} = \frac{a_{3j} - l_{31}u_{1j} - l_{32}u_{2j}}{l_{33}}, \; j = 4,5,\ldots,n$$

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

ด้วยการตรวจสอบในผลลัพธ์ที่ได้ เราสามารถสรุปเป็นชุดสมการในการคำนวณได้ดังนี้

$$l_{i1} = a_{i1}, \quad for\ i = 1,2,\ldots,n$$

$$u_{ij} = \frac{a_{1j}}{l_{11}}, \quad for\ j = 2,3,\ldots,n$$

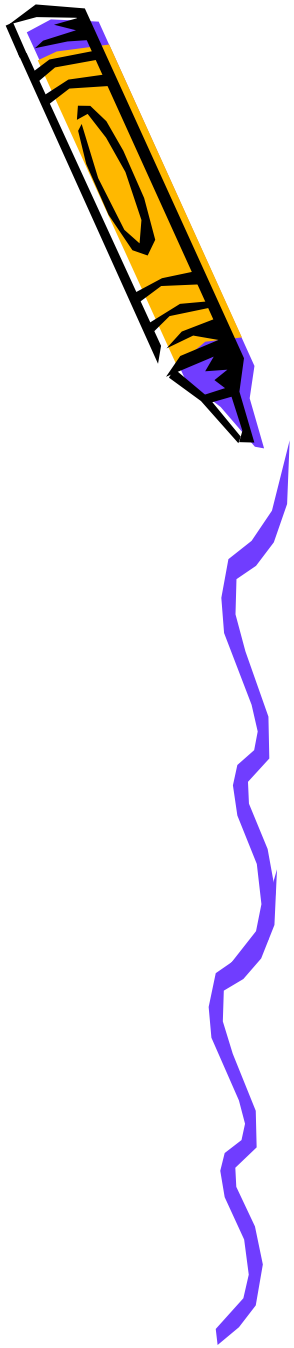
$$For\ j = 2,3,\ldots,n-1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad for\ i = j, j+1,\ldots,n$$

$$u_{jk} = \frac{a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik}}{l_{jj}}, \quad for\ k = j+1, j+2,\ldots,n$$

$$l_{nn} = a_{nn} - \sum_{k=1}^{n-1} l_{nk}u_{kn}$$

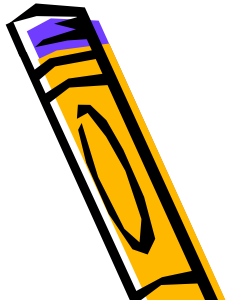
Crout Decomposition



และ Pseudo Code สามารถเขียนได้เป็น

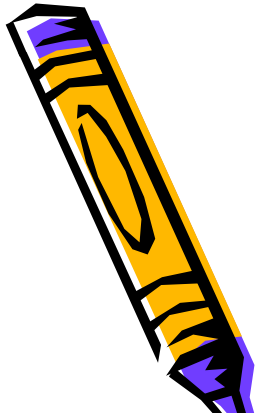
```
DOFOR j = 2 to n
   a(1,j)=a(1,j)/a(1,1)
ENDDO
DOFOR j = 2 to n-1
   DOFOR i = j to n
      sum=0;
      DOFOR k = 1 to j-1
         sum=sum+a(i,k)*a(k,j)
      ENDDO
      a(i,j)=a(i,j)-sum
   ENDDO
   DOFOR k = j+1 to n
      sum=0;
      DOFOR i = 1 to j-1
         sum=sum+a(j,i)*a(i,k)
      ENDDO
      a(j,k)=(a(j,k)-sum)/a(j,j)
   ENDDO
ENDDO
sum=0
DOFOR k = 1 to n-1
   sum=sum+a(n,k)*a(k,n)
ENDDO
a(n,n)=a(n,n)-sum
```

Crout Decomposition



สังเกตว่า Algorithm ประกอบด้วย Loop ที่กะทัดรัด นอกเหนือจากนั้นแล้วลักษณะของ Algorithm จะเป็น In-place กล่าวคือทั้ง Upper Diagonal และ Lower Diagonal Matrix สามารถบรรจุอยู่ใน Matrix เดียวกันและเข้าไปแทนที่ Element ใน Matrix เดิม เนื่องจากค่า Coefficient ของ Matrix **A** จะถูกใช้เพียงแค่ครั้งเดียว ดังนั้นนอกจากจะรวดเร็วแล้ว Algorithm ยังใช้หน่วยความจำน้อย ภาคผนวกแสดงโปรแกรม MATLAB แต่ในกรณีนี้จะแยกแต่ละ Matrix ออกจากกัน เพื่อความสะดวกในการศึกษาการทำงาน

Example 8.6



Example 8.6 จงใช้ Crout Decomposition เพื่อทำ LU Decomposition ของสมการ

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ -8 \\ 16 \end{bmatrix}$$

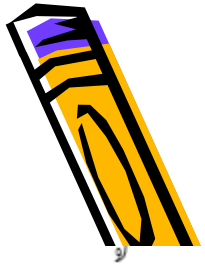
$$l_{i1} = a_{i1}, \quad for\ i = 1,2,\ldots,n$$

$$u_{ij} = \frac{a_{1j}}{l_{11}}, \quad for\ j = 2,3,\ldots,n$$

ผลการ Run Program ได้ดังนี้

```
ans =
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000         0         0         0         0         0
    0.3000         0         0         0         0         0

ans =
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000    7.0033         0         0    1.0000   -0.0419
    0.3000   -0.1900         0         0         0         0

ans =
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000    7.0033         0         0    1.0000   -0.0419
    0.3000   -0.1900   10.0120         0         0    1.0000
```

Crout Decomposition



ในการนำ Crout Decomposition ไปแก้สมการ Linear Equation จะต้องมีขบวนการ Substitution และในกรณีนี้เราจะต้องคำนวณ Vector **D** จาก Vector **C** ด้วย หลังจากเราทำ LU Decomposition แล้ว

จากสมการ $\mathbf{LD} = \mathbf{C}$ ที่กล่าวในหัวข้อก่อน หรือในกรณีของสมการ 4 Unknown

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

เราสามารถสรุปจากการคูณกันของ Matrix ได้คำตอบดังนี้

$$d_1 = \frac{c_1}{l_{11}}; \quad d_i = \frac{c_i - \sum_{j=1}^{i-1} l_{ij} d_j}{l_{ii}}, \quad i = 2,3,...,n$$

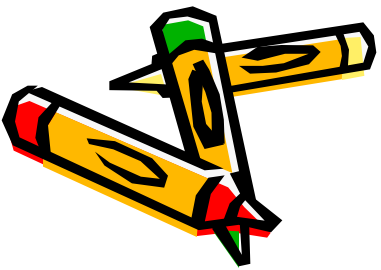
Crout Decomposition

และค่าของ x_i สามารถคำนวณได้จากวิธีการ Back Substitution จากสมการ $\mathbf{UX} = \mathbf{D}$ หรือ

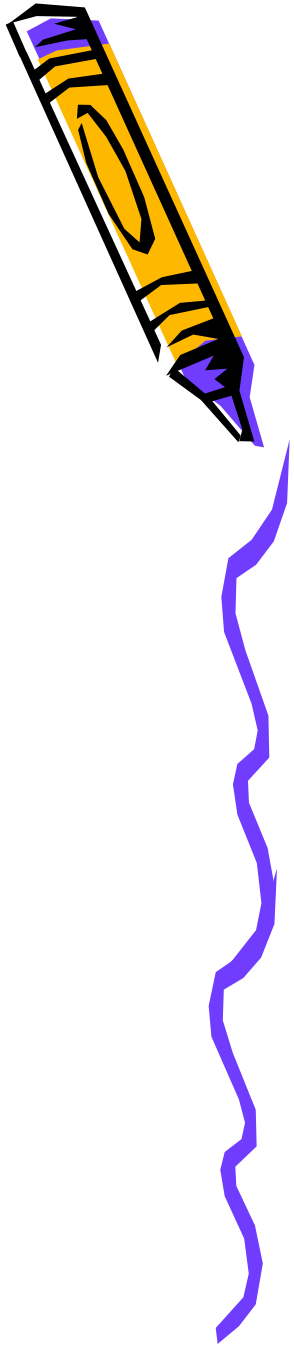
$$\begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

ได้ดังนี้

$$x_n = d_n; \quad x_i = d_i - \sum_{j=i+1}^{n} u_{ij} x_j, \quad i = n-1, n-2, \ldots, 1$$



Example 8.7



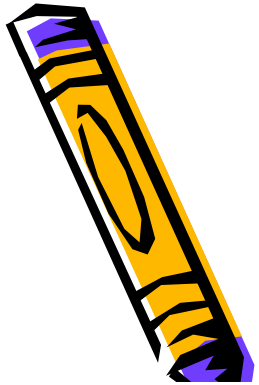
Example 8.7 จงใช้ Crout Decomposition เพื่อแก้สมการ

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ -8 \\ 16 \end{bmatrix}$$

จากโปรแกรมในภาคผนวก เราได้ดังนี้

Example 8.7

```
Calculate LU Decomposition, Row&Col = 1 ; ans = [L U]
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000         0         0         0         0         0
    0.3000         0         0         0         0         0
Calculate LU Decomposition, Row&Col = 2 ; ans = [L U]
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000    7.0033         0         0    1.0000   -0.0419
    0.3000   -0.1900         0         0         0         0
Calculate LU Decomposition, Row&Col = 3 ; ans = [L U]
    3.0000         0         0    1.0000   -0.0333   -0.0667
    0.1000    7.0033         0         0    1.0000   -0.0419
    0.3000   -0.1900   10.0120         0         0    1.0000
Find D from L and C ; ans = [L D C]
    3.0000         0         0    4.0000   12.0000
    0.1000    7.0033         0   -1.1994   -8.0000
    0.3000   -0.1900   10.0120    1.4555   16.0000
Find X from U and D ; ans = [U X D]
    1.0000   -0.0333   -0.0667    4.0591    4.0000
         0    1.0000   -0.0419   -1.1385   -1.1994
         0         0    1.0000    1.4555    1.4555
```



ในการนำ Crout Algorithm มาดัดแปลงหา Inverse ของ Matrix เราสามารถทำได้โดยใช้วิธีที่กล่าวมาในตอนต้น คือทำการหาทีละ Column ในการนี้เราทำการหา Upper และ Lower Diagonal Matrix ครั้งเดียว แต่การหา **D** นั้นจะต้องกระทำแยกและค่อยทำ Back-Substitution หาค่า **X** ที่เป็น Solution ของแต่ละ Column ของ Inverse Matrix

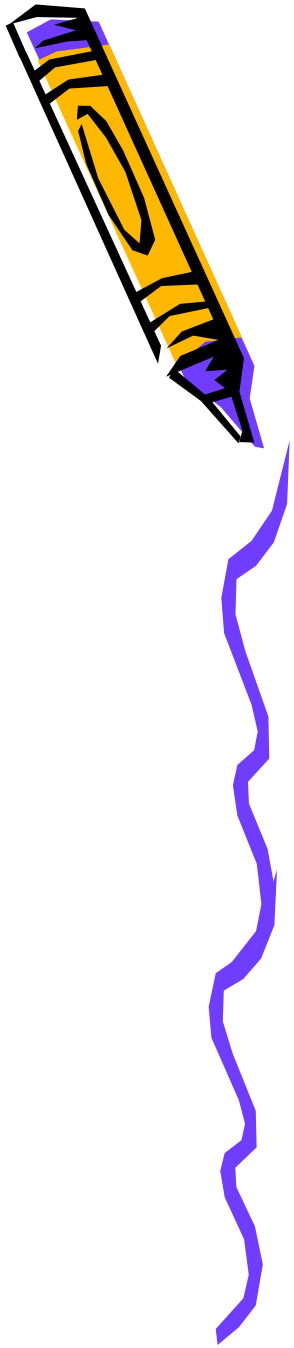
กรรมวิธีที่กล่าวมาทั้งหมด สามารถนำมาปรับปรุงให้มีประสิทธิภาพเพิ่มขึ้นได้อีก ถ้า Matrix มีคุณสมบัติพิเศษ อย่างเช่นในกรณีของ Sparse Matrix หรือใน Banded System ที่ Matrix เป็น Multi-diagonal Matrix ซึ่งเรามี Algorithm ที่มีประสิทธิภาพสูงเฉพาะสำหรับ Matrix ประเภทนี้ หรือใน Matrix ที่เป็น Symmetric Matrix ซึ่งในกรณีหลังนี้เรามี Algorithm ที่ชื่อ Cholsky Decomposition ที่รวดเร็วกว่า Crout Decomposition อย่างไรก็ตาม รายละเอียดของ Algorithm เหล่านี้จะเกินเนื้อหาของวิชานี้

Method	จำนวนสูงสุดของ Unknown	Stability	Precision	Breadth of Application	Program	Comments
Graphical	2	-	Poor	Limited	-	อาจใช้เวลามาก เมื่อเทียบกับวิธี Numerical
Cramer's Rule	3	-	ได้รับผลจาก Round-Off Error	Limited	-	ใช้การคำนวณมากเกินไป ถ้ามีมากกว่า 3 สมการ
Algebraic Elimination	3	-	ได้รับผลจาก Round-Off Error	Limited	-	
Gauss Elimination with Pivoting	100	-	ได้รับผลจาก Round-Off Error	General	ยากปานกลาง	
Gauss-Jordan with Partial Pivoting	100	-	ได้รับผลจาก Round-Off Error	General	ยากปานกลาง	สามารถนำมาใช้คำนวณ Matrix Inverse ได้ดี
LU Decomposition	100	-	ได้รับผลจาก Round-Off Error	General	ยากปานกลาง	เป็นวิธี Elimination ที่นิยมมากที่สุด
Gauss-Seidel	1000	อาจจะไม่ Converge ถ้าไม่เป็น Diagonally	Excellent	เหมาะสมเฉพาะกับระบบที่เป็น Diagonally	ง่าย	

Homework 9, Chapter 10

- DOWNLOAD
- คำนวณแนะนำให้เขียนโปรแกรม หรือใช้ MATLAB หรือใช้ Spreadsheet(Excel)
- หยุด 2 สัปดาห์
  - การบ้าน ส่งต้นชั่วโมง พุธ 20 เมษายน

End of Chapter 10



- Next Week (Wk 13)
  - No Class วันจักรี
- Following Week (Wk 14)
  - No Class วันสงกรานต์
- Week 15; Chapter 11
  - April 20
  - Numerical Differentiation and Integration