9장 Gauss 소거법

9.1 소규모의 방정식을 풀기

9.2 순수 Gauss 소거법

9.3 피봇팅

9.4 삼중대각 시스템

9장 Gauss 소거법

어떤 원리에 의해 다음과 같은 MATLAB 명령어가 수행되는가?

Command Window

File Edit View Web Window Help

```
>> x=A\b

>> x=inv(A)*b
```

Ready

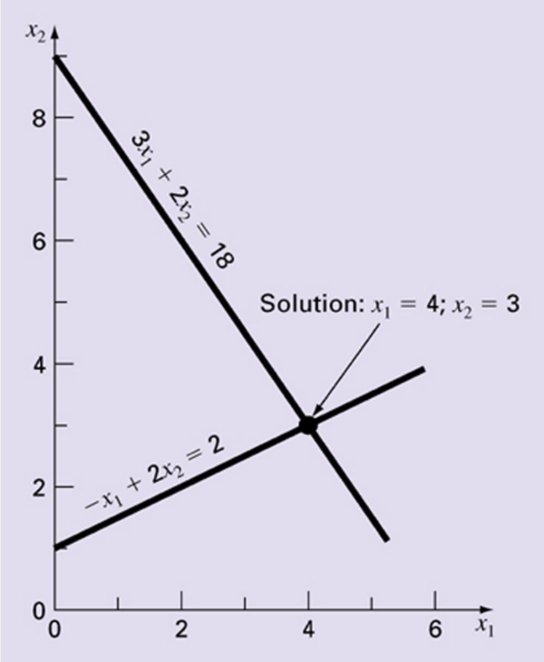
9.1 소규모의 방정식을 풀기 (1/6)

컴퓨터를 필요로 하지 않고 소규모 연립방정식 ($n \le 3$)에 적합한 방법

– 도식적 방법, Cramer 공식, 미지수 소거법

▪ 도식적인 방법

$$\begin{aligned} 3x_1 + 2x_2 &= 18 \\ -x_1 + 2x_2 &= 2 \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} x_2 &= -\frac{3}{2}x_1 + 9 \\ x_2 &= \frac{1}{2}x_1 + 1 \end{aligned}$$

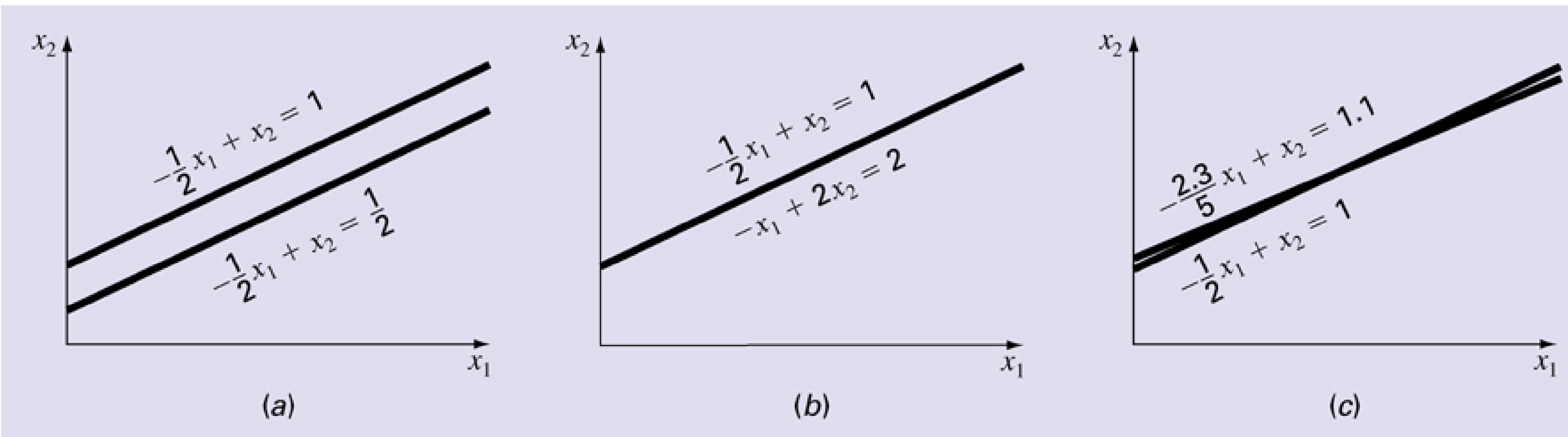


두 연립선형대수방정식의 도식적인 해 (교점이 해를 나타냄)

9.1 소규모의 방정식을 풀기 (2/6)

- *특이* 평행선 → 해가 없음

  두 선이 일치함 → 해가 무한히 많음
- *불량조건* 특이에 가까워 반올림오차에 매우 민감함



특이 시스템과 불량조건 시스템의 도식적 표현

(a) 해가 없음, (b) 해가 무한히 많음, (c) 해를 식별하기 어려움.

9.1 소규모의 방정식을 풀기 (3/6)

■ 행렬식과 Cramer 공식

세 개의 방정식에 대한 행렬식을 고려하자.

$$[A]\{x\}=\{b\} \qquad \text{여기서} \qquad [A]=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

이 시스템의 *행렬식*은 계수행렬로부터 구성되는데 다음과 같다.

$$D=\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\overbrace{\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}}^{\text{minor}} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

$$= a_{11}(a_{22}a_{33}-a_{23}a_{32}) - a_{12}(a_{21}a_{33}-a_{23}a_{31}) + a_{13}(a_{21}a_{32}-a_{22}a_{31})$$

예제 9.1

Q. 다음 시스템에 대하여 행렬식을 구하라.

① $\begin{cases} 3x_1 + 2x_2 = 18 \\ -x_1 + 2x_2 = 2 \end{cases}$
② $\begin{cases} -\dfrac{1}{2}x_1 + x_2 = 1 \\ -\dfrac{1}{2}x_1 + x_2 = \dfrac{1}{2} \end{cases}$
③ $\begin{cases} -\dfrac{1}{2}x_1 + x_2 = 1 \\ -x_1 + 2x_2 = 2 \end{cases}$
④ $\begin{cases} -\dfrac{2.3}{5}x_1 + x_2 = 1.1 \\ -\dfrac{1}{2}x_1 + x_2 = 1 \end{cases}$

풀이)

① $D = \begin{vmatrix} 3 & 2 \\ -1 & 2 \end{vmatrix} = 3(2) - 2(-1) = 8$

② $D = \begin{vmatrix} -\dfrac{1}{2} & 1 \\ -\dfrac{1}{2} & 1 \end{vmatrix} = -\dfrac{1}{2}(1) - 1\left(\dfrac{-1}{2}\right) = 0$

특이조건

③ $D = \begin{vmatrix} -\dfrac{1}{2} & 1 \\ -1 & 2 \end{vmatrix} = -\dfrac{1}{2}(2) - 1(-1) = 0$

특이조건

④ $D = \begin{vmatrix} -\dfrac{1}{2} & 1 \\ -\dfrac{2.3}{5} & 1 \end{vmatrix} = -\dfrac{1}{2}(1) - 1\left(\dfrac{-2.3}{5}\right) = -0.04$

불량조건

9.1 소규모의 방정식을 풀기 (4/6)

- 특이 시스템 → 행렬식 = 0
- 불량조건 시스템 → 행렬식이 0에 가까움

- Cramer 공식

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D}$$

예제 9.2 (1/2)

Q. Cramer 공식으로 다음의 연립방정식을 풀어라.

$$
\begin{aligned}
0.3x_1 &+ 0.52x_2 &+ x_3 &= -0.01 \\
0.5x_1 &+ x_2 &+ 1.9x_3 &= 0.67 \\
0.1x_1 &+ 0.3x_2 &+ 0.5x_3 &= -0.44
\end{aligned}
$$

풀이)

$$
D = 0.3\begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} - 0.52\begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1\begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.3 \end{vmatrix} = -0.0022
$$

예제 9.2 (2/2)

풀이)

$$x_1 = \frac{\begin{vmatrix} -0.01 & 0.52 & 1 \\ 0.67 & 1 & 1.9 \\ -0.44 & 0.3 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.03278}{-0.0022} = -14.9$$

$$x_2 = \frac{\begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.0649}{-0.0022} = -29.5$$

$$x_3 = \frac{\begin{vmatrix} 0.3 & 0.52 & -0.01 \\ 0.5 & 1 & 0.67 \\ 0.1 & 0.3 & -0.44 \end{vmatrix}}{-0.0022} = \frac{-0.04256}{-0.0022} = -19.8$$

Cramer 공식은 $n > 3$ 이면 *비현실적* 이다.

9.1 소규모의 방정식을 풀기 (5/6)

▪ 미지수 소거법

두 개의 방정식으로 구성된 경우를 고려해 보자.

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

상수를 곱하면

$$a_{21}a_{11}x_1 + a_{21}a_{12}x_2 = a_{21}b_1$$

$$a_{11}a_{21}x_1 + a_{11}a_{22}x_2 = a_{11}b_2$$

뺄셈으로 x_1을 소거하면 $a_{11}a_{22}x_2 - a_{21}a_{12}x_2 = a_{11}b_2 - a_{21}b_1$

따라서 $x_2 = \dfrac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}$ 이 결과를 대입하면 $x_1 = \dfrac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{21}a_{12}}$

9.1 소규모의 방정식을 풀기 (6/6)

미지수 소거법은 Cramer 공식을 그대로 따른 것이다.

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{21}a_{12}}$$

$$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{21}a_{12}}$$

이 기법은 컴퓨터를 이용하도록 공식화되어 있어 프로그램 작성이 용이하다.

9.2 순수 Gauss 소거법 (1/14)

- Gauss 소거법의 두 단계: 전진소거와 후진대입

① 하나의 방정식에
오직 하나의 미지수만 나타나도록
방정식을 조작한다.

② 그 미지수를 풀고,
후진대입을 통해
나머지 미지수도 결정한다.

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array}\right]$$

(*a*) Forward elimination

$$\downarrow$$

$$x_3 = b''_3/a''_{33}$$

$$x_2 = (b'_2 - a'_{23}x_3)/a'_{22}$$

$$x_1 = (b_1 - a_{13}x_3 - a_{12}x_2)/a_{11}$$

(*b*) Back substitution

9.2 순수 Gauss 소거법 (2/14)

- 알고리즘을 개발하면 대규모의 방정식으로 확장이 가능하다.

- Gauss 소거법은 가장 기본적인 알고리즘이다.

- "순수" Gauss 소거법은 0으로 나누는 경우를 극복하지 못한 방법이다.

9.2 순수 Gauss 소거법 (3/14)

- n 개의 미지수를 가진 방정식을 다루어 보자.

피봇원소 (첫 번째 방정식의 미지수의 계수)

$$
\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_1 & + & a_{13}x_1 & + & \cdots & + & a_{1n}x_1 & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_2 & + & \cdots & + & a_{2n}x_2 & = & b_2 \\
 & & \vdots & & \vdots & & & & & & \\
a_{n1}x_1 & + & a_{n2}x_2 & + & a_{n3}x_3 & + & \cdots & + & a_{nn}x_n & = & b_n
\end{array}
$$

피봇방정식

(첫 번째 미지수에 대한)

9.2 순수 Gauss 소거법 (4/14)

단계 ①: 미지수의 전진소거
(방정식을 *상삼각*시스템으로 바꿈)

두 번째에서 n번째 방정식까지 미지수 x_1을 소거한다.

첫 번째 식에 a_{21}/a_{11}을 곱하면

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \frac{a_{21}}{a_{11}}a_{13}x_3 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1$$

두 번째 식에서 빼면

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n = b_2 - \frac{a_{21}}{a_{11}}b_1$$

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

9.2 순수 Gauss 소거법 (5/14)

나머지 방정식에서도 대해 반복하면

$$
\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
 & & a'_{22}x_2 & + & a'_{23}x_3 & + & \cdots & + & a'_{2n}x_n & = & b'_2 \\
 & & a'_{32}x_2 & + & a'_{33}x_3 & + & \cdots & + & a'_{3n}x_n & = & b'_3 \\
 & & \vdots & & \vdots & & & & & & \\
 & & a'_{n2}x_2 & + & a'_{n3}x_3 & + & \cdots & + & a'_{nn}x_n & = & b'_n
\end{array}
$$

두 번째 피봇방정식을 이용하여 미지수 x_2를 소거하면

$$
\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
 & & a'_{22}x_2 & + & a'_{23}x_3 & + & \cdots & + & a'_{2n}x_n & = & b'_2 \\
 & & & & a''_{33}x_3 & + & \cdots & + & a''_{3n}x_n & = & b''_3 \\
 & & & & \vdots & & & & & \vdots & \\
 & & & & a''_{n3}x_3 & + & \cdots & + & a''_{nn}x_n & = & b''_n
\end{array}
$$

9.2 순수 Gauss 소거법 (6/14)

나머지 피봇방정식을 이용하여 계속적으로 소거를 수행하면 상삼각행렬 시스템을 구성할 수 있다.

$$
\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
 & & a'_{22}x_2 & + & a'_{23}x_3 & + & \cdots & + & a'_{2n}x_n & = & b'_2 \\
 & & & & a''_{33}x_3 & + & \cdots & + & a''_{3n}x_n & = & b''_3 \\
 & & & & & \ddots & & & & \vdots & \\
 & & & & & & & & a_{nn}^{(n-1)}x_n & = & b_n^{(n-1)}
\end{array}
$$

9.2 순수 Gauss 소거법 (7/14)

- 단계 ②: 후진대입

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

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}} \qquad (i = n-1, n-2, \ldots, 1)$$

예제 9.3 (1/2)

Q. Gauss 소거법을 이용하여 방정식을 풀어라.
(정해는 x_1 = 3, x_2 = -2.5, 그리고 x_3 = 7)

$$
\begin{aligned}
3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\
0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\
0.3x_1 - 0.2x_2 + 10x_3 &= 71.4
\end{aligned}
$$

풀이) 전진소거를 수행하면

$$
\begin{aligned}
3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\
7.00333x_2 - 0.293333x_3 &= -19.5617 \\
-0.190000x_2 + 10.0200x_3 &= 70.6150
\end{aligned}
$$

$$
\begin{aligned}
3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\
7.00333x_2 - 0.293333x_3 &= -19.5617 \\
10.0120x_3 &= 70.0843
\end{aligned}
$$

예제 9.3 (2/2)

풀이) 후진대입하여 해를 구하면

$$x_3 = \frac{70.0843}{10.0120} = 7.00003$$

$$x_2 = \frac{-19.5617 + 0.293333(7.00003)}{7.00333} = -2.50000$$

$$x_1 = \frac{7.85 + 0.1(-2.50000) + 0.2(7.00003)}{3} = 3.00000$$

결과를 확인하면

$$3(3) - 0.1(-2.5) - 0.2(7.00003) = 7.84999 \cong 7.85$$

$$0.1(3) + 7(-2.5) - 0.3(7.00003) = -19.30000 \cong -19.3$$

$$0.3(3) - 0.2(-2.5) + 10(7.00003) = 71.4003 \cong 71.4$$

9.2 순수 Gauss 소거법 (8/14)

[순수 Gauss 소거법을 수행하는 M-파일]

```
function x = GaussNaive(A,b)
% GaussNaive (A,b):
%   Gauss elimination without pivoting
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector
[m,n] = size(A);
if m ~= n, error('Matrix A must be square'); end
nb = n+1;
Aug =[A b];
```

9.2 순수 Gauss 소거법 (8/14)

[순수 Gauss 소거법을 수행하는 M-파일]

```
% forward elimination
for k = 1:n-1
   for i = k+1:n
      factor = Aug(i,k)/Aug(k,k);
      Aug(i,k:nb) = Aug(i,k:nb)-factor*Aug(k,k:nb);
   end
end
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
   x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

■ 연산 횟수

- 수행시간 ~ 부동소수점 연산(*flops*)의 횟수
- 몇 개의 항들을 정의하면

$$\sum_{i=1}^{m} cf(i) = c\sum_{i=1}^{m} f(i) \qquad \sum_{i=1}^{m}\{f(i)+g(i)\} = \sum_{i=1}^{m} f(i) + \sum_{i=1}^{m} g(i)$$

$$\sum_{i=1}^{m} 1 = 1+1+1+\cdots+1 = m \qquad \sum_{i=k}^{m} 1 = m-k+1$$

$$\sum_{i=1}^{m} i = 1+2+3+\cdots+m = \frac{m(m+1)}{2} = \frac{m^2}{2} + O(m)$$

$$\sum_{i=1}^{m} i^2 = 1^2+2^2+3^2+\cdots+m^2 = \frac{m(m+1)(2m+1)}{6} = \frac{m^3}{3} + O(m^2)$$

여기서 $O(m^n)$ = **크기가** m^n **차수와 그보다 낮은 차수의 항**

9.2 순수 Gauss 소거법 (10/14)

- **순수 Gauss 소거법의 M-파일에 대하여**

외부 루프가 k = 1에서 시작하므로,

내부 루프의 한계는 i = 2 에서부터 n까지다.

따라서 내부 루프의 반복 횟수는 $\sum_{i=2}^{n} 1 = n - 2 + 1 = n - 1$

- (n − 1)번 반복하는 내부 루프에 대해서
  - 나눗셈 한 번
  - 각각의 열 요소에 대해 2 에서 nb까지 곱셈 (n + 1까지 n 번의 곱셈)
  - 마찬가지로 n 번 뺄셈
  - 합하면 n + 1 번 곱셈/나눗셈과 n 번 뺄셈
  - 외부 루프를 한 번 지나는 것에 대해

    전체적으로 (n − 1)(n + 1)번 곱셈/나눗셈과 (n − 1)(n)번 뺄셈

9.2 순수 Gauss 소거법 (11/14)

- 요약하면

외부 루프 k	내부 루프 i	덧셈/뺄셈 연산횟수	곱셈/나눗셈 연산횟수
1	2, n	$(n-1)(n)$	$(n-1)(n+1)$
2	3, n	$(n-2)(n-1)$	$(n-2)(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	$k+1, n$	$(n-k)(n+1-k)$	$(n-k)(n+k-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	n, n	$(1)(2)$	$(1)(3)$

9.2 순수 Gauss 소거법 (12/14)

따라서 소거를 위한 전체 덧셈/뺄셈 연산횟수는

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

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

결과적으로 다음과 같다.

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

유사한 해석을 곱셈/나눗셈에 대해서 수행하면

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

9.2 순수 Gauss 소거법 (13/14)

최종적으로

$$\frac{2n^3}{3} + O(n^2)$$

O(n^2) 이하의 항들은 n이 증가할수록 무시됨을 유의하라..

후진대입에 대하여

덧셈/뺄셈 연산횟수 = n(n − 1)/2

곱셈/나눗셈 연산횟수 = n(n + 1)/2

따라서 합은 $n^2 + O(n)$

9.2 순수 Gauss 소거법 (14/14)

순수 Gauss 소거법에 소요되는 전체 연산횟수는 다음과 같다.

$$\underbrace{\frac{2n^3}{3}+O(n^2)}_{\text{Forward elimination}} + \underbrace{n^2+O(n)}_{\text{Back substitution}} \xrightarrow{\text{as n increases}} \frac{2n^3}{3}+O(n^2)$$

① 시스템이 커질수록 연산시간이 크게 증가한다.

② 대부분의 연산은 소거단계에서 발생한다.

n	전진소거	후진대입	전체 연산횟수	**2*n*³/3**	소거의 비율 %
10	705	100	805	667	87.58%
100	671550	10000	681550	666667	98.53%
1000	6.67×10^8	1×10^6	6.68×10^8	6.67×10^8	99.85%

9.3 피봇팅 (1/3)

$$\begin{array}{rcrcrcr} & & 2x_2 & + & 3x_3 & = & 8 \\ 4x_1 & + & 6x_2 & + & 7x_3 & = & -3 \\ 2x_1 & - & 3x_2 & + & 6x_3 & = & 5 \end{array}$$

→ 순수 Gauss 소거법의 정규화에서
0으로 나누는 나눗셈 발생!

피봇 원소가 0에 가까우면 어떤 일이 일어나는가?

☹ 피봇 원소의 크기가 다른 원소에 비해 작으면 반올림오차가 개입!

처방

☑ 각각의 행을 정규화하기 전에 해당 열에서 계수가 가장 큰 것을 찾는다.

☑ 가장 큰 원소가 피봇 원소가 되도록 순서를 바꾼다

⇒ *부분 피봇팅*

☑ 열과 행에서 가장 큰 원소를 찾아 순서를 바꾼다

⇒ *완전 피봇팅* → 드물게 사용됨

예제 9.4 (부분 피봇팅) (1/3)

Q. Gauss 소거법을 이용하여 다음의 방정식을 풀어라.

$$0.0003x_1 + 3.0000x_2 = 2.0001$$
$$1.0000x_1 + 1.0000x_2 = 1.0000$$

첫 번째 피봇 원소는 a_{11} = 0.0003이며, 0에 매우 가깝다.
그래서 방정식의 순서를 바꾸는 부분 피봇팅을 취한다.
정해는 x_1 = 1/3과 x_2 = 2/3이다.

예제 9.4 (부분 피봇팅) (2/3)

풀이) 첫 번째 식에 1/(0.0003)을 곱하면

$$\rightarrow \quad x_1 + 10{,}000x_2 = 6667$$

두 번째 식에서 이 식을 빼면 결과는

$$\rightarrow \quad -9999x_2 = -6666 \quad \rightarrow \quad x_2 = 2/3$$

따라서

$$x_1 = \frac{2.0001 - 3(2/3)}{0.0003}$$

유효숫자 수	x_2	x_1	x_1의 백분율 상대오차의 절대값
3	0.667	-3.33	1099
4	0.6667	0.0000	100
5	0.66667	0.30000	10
6	0.666667	0.330000	1
7	0.6666667	0.3330000	0.1

예제 9.4 (부분 피봇팅) (3/3)

- 거의 같은 두 수 사이의 뺄셈으로 인해 결과가 유효숫자 수에 매우 민감하다.

순서를 바꾸어서 방정식을 풀면 (= 부분 피봇팅)

$$1.0000x_1 + 1.0000x_2 = 1.0000$$
$$0.0003x_1 + 3.0000x_2 = 2.0001$$

→ x_2 = 2/3 그리고 $x_1 = \dfrac{1-(2/3)}{1}$

유효숫자 수	x_2	x_1	x_1의 백분율 상대오차의 절대값
3	0.667	0.333	0.1
4	0.6667	0.3333	0.01
5	0.66667	0.33333	0.001
6	0.666667	0.333333	0.0001
7	0.6666667	0.3333333	0.00001

9.3 피봇팅 (2/3)

[부분 피봇팅이 포함된 Gauss 소거법을 위한 M-파일]

Untitled

File Edit View Text Debug Breakpoints Web Window Help

```
function x = GaussPivot(A,b)
% GaussPivot (A,b):
%   Gauss elimination with partial pivoting
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector
[m,n] = size(A);
if m ~= n, error('Matrix A must be square'); end
nb = n+1;
Aug =[A b];
```

Ready

9.3 피봇팅 (2/3)

[부분 피봇팅이 포함된 Gauss 소거법을 위한 M-파일]

```
% forward elimination
for k = 1:n-1
    % partial pivoting
    [big, i] = max(abs(Aug(k:n,k)));
    ipr = i + k -1;
    if ipr  ~= k
        % pivot the row
        Aug([k,ipr],:) = Aug([ipr,k],:);
    end
```

9.3 피봇팅 (2/3)

[부분 피봇팅이 포함된 Gauss 소거법을 위한 M-파일]

```
    for i = k+1:n
    factor = Aug(i,k)/Aug(k,k);
    Aug(i,k:nb) = Aug(i,k:nb)-factor*Aug(k,k:nb);
    end
end
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
    x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

9.3 피봇팅 (3/3)

Command Window

File Edit View Web Window Help

```
>> y = [1; 3; 5; 2; 4];
>> [ymax, imax] =max(y)
ymax =
     5
imax =
     3


>> A=[0.0003 3; 1 1]; b=[2.0001; 1];
>> x = GaussPivot(A,b)
x =
    0.3333
    0.6667
```

Ready

9.4 삼중대각 시스템 (1/3)

띠의 폭이 3인 삼중대각 시스템을 고려하자.

$$
\begin{bmatrix}
f_1 & g_1 & & & & & & \\
e_2 & f_2 & g_2 & & & & & \\
 & e_3 & f_3 & g_3 & & & & \\
 & & \cdot & \cdot & \cdot & & & \\
 & & & \cdot & \cdot & \cdot & & \\
 & & & & \cdot & \cdot & \cdot & \\
 & & & & & e_{n-1} & f_{n-1} & g_{n-1} \\
 & & & & & & e_n & f_n
\end{bmatrix}
\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{Bmatrix}
=
\begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdot \\ \cdot \\ \cdot \\ r_{n-1} \\ r_n \end{Bmatrix}
$$

예제 9.5 (삼중대각 시스템의 해) (1/2)

Q. 다음의 삼중대각 시스템의 해를 구하라.

$$\begin{bmatrix} 2.04 & -1 & & \\ -1 & 2.04 & -1 & \\ & -1 & 2.04 & -1 \\ & & -1 & 2.04 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 40.8 \end{Bmatrix}$$

예제 9.5 (삼중대각 시스템의 해) (2/2)

풀이)

전진소거를 통해

$$\begin{bmatrix} 2.04 & -1 & & \\ & 1.550 & -1 & \\ & & 1.395 & -1 \\ & & & 1.323 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 40.8 \\ 20.8 \\ 14.221 \\ 50.996 \end{Bmatrix}$$

후진대입으로 해를 구하면

$$x_4 = \frac{r_4}{f_4} = \frac{50.996}{1.323} = 38.545$$

$$x_3 = \frac{r_3 - g_3 x_4}{f_3} = \frac{14.221 - (-1)38.545}{1.395} = 37.832$$

$$x_2 = \frac{r_2 - g_2 x_3}{f_2} = \frac{20.800 - (-1)37.832}{1.550} = 37.832$$

$$x_1 = \frac{r_1 - g_1 x_2}{f_1} = \frac{40.800 - (-1)37.832}{2.040} = 38.545$$

9.4 삼중대각 시스템 (2/3)

[상삼각 시스템을 풀기 위한 M-파일]

```
function x = Tridiag(e, f, g, r)
% Tridiag(e, f, g, r):
%    Tridiagonal system solver
% input:
%    e = subdiagonal vector
%    f = diagonal vector
%    g = superdiagonal vector
%    r = right hand side vector
% output
%    x = solution vector
```

9.4 삼중대각 시스템 (2/3)

[상삼각 시스템을 풀기 위한 M-파일]

```
n = length(f);
% forward elimination
for k = 2:n
   factor = e(k)/f(k-1);
   f(k) = f(k) - factor*g(k-1);
   r(k) = r(k) - factor*r(k-1);
end
% back substitution
disp(f); disp(r)          % check modified coefficients
x(n) = r(n)/f(n);
for k = n-1:-1:1
   x(k) = (r(k)-g(k)*x(k+1))/f(k);
end
```

9.4 삼중대각 시스템 (3/3)

Command Window

File Edit View Web Window Help

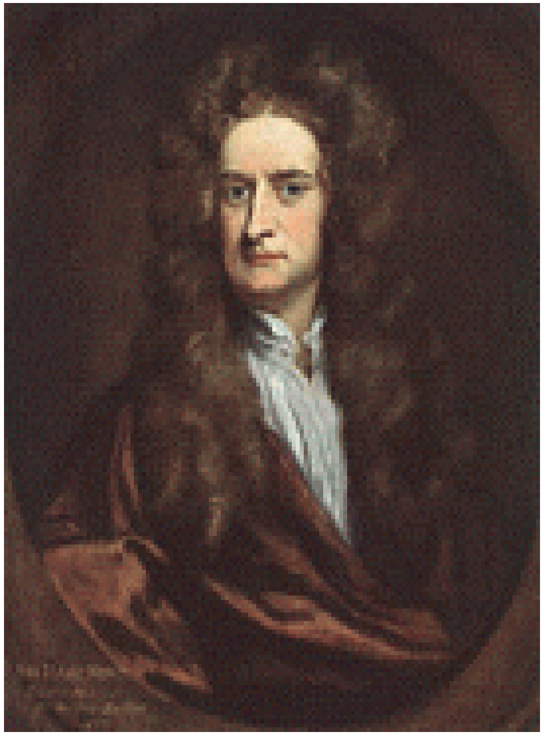
```
>>
>> e=[0; -1; -1; -1];
>> f=[2.04; 2.04; 2.04; 2.04];
>> g=[-1; -1; -1; 0];
>> r=[40.8; 0.8; 0.8; 40.8];
>> x = Tridiag(e, f, g, r)
    2.0400    1.5498    1.3948    1.3230
   40.8000   20.8000   14.2211   50.9961
x =
   38.5449   37.8317   37.8317   38.5449
```

- 대부분의 상삼각 시스템에서는 피봇팅이 필요하지 않으나 드물게 요구되는 경우도 있다.
- 상삼각 시스템에 소요되는 계산 노력~ n (참고로 Gauss 소거법 ~ n^3)

파스칼(Blaise Pascal 1623~1662)

- 1635년 12세 때 유클리드기하학 정리 추론
- 1639년 16세 때 유명 수학자들로부터 주목
- 1642년 계산기를 발명
- 1647년 유체정역학 ‘파스칼의 원리’를 발견

뉴턴(Sir Isaac Newton 1642~1727)



- 1669년 27세에 루카시안교수직을 받아 강의
- 1668년, 1671년 반사망원경을 제작
- 1672년 왕립협회 회원
- 뉴턴의 3개의 대발견: 빛의 스펙트럼, 만유인력의 법칙, 미적분과 3가지 운동법칙

라플라스: '운동법칙' 은 인간의 가장 위대한 작품

뉴턴

프린키피아의 결론: “태양, 행성들, 그리고 혜성들의 가장 아름다운 체계는 이지적이며 능력 있는 계획과 통치로부터만 나올 수 있다.… 이 존재는, 세계의 영혼으로서가 아니라 만유의 주로서 모든 것을 다스린다. 그리고 그 통치 사실 때문에 그분은 주 하나님이라고 일컬어진다.”

패러디 (Michael Faraday 1791~1867)

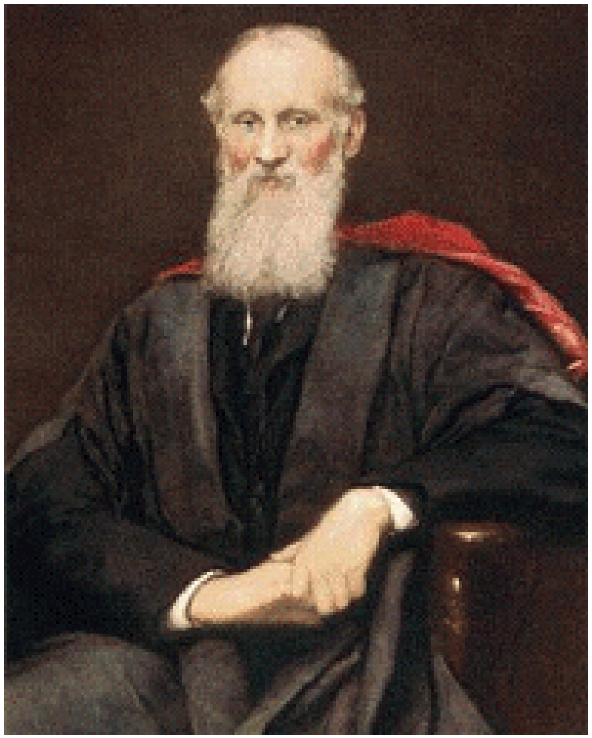
- 1813년 화학자인 데이비의 왕립연구소 조수
- 1824년 왕립학회회원
- 1833년 왕립연구소 화학교수
- 1823년 염소가스를 액화 성공
- 1825년 벤젠을 발견
- 1833년 ‘패러디 법칙’을 발견
- 1845년 ‘패러디 효과’와 반자성체의 발견

아인슈타인: 갈릴레이와 뉴턴과 어깨를 나란히 견줄 수 있는 역사상 가장 위대한 과학자 세분 중 또 한 분은 패러디뿐

패러디

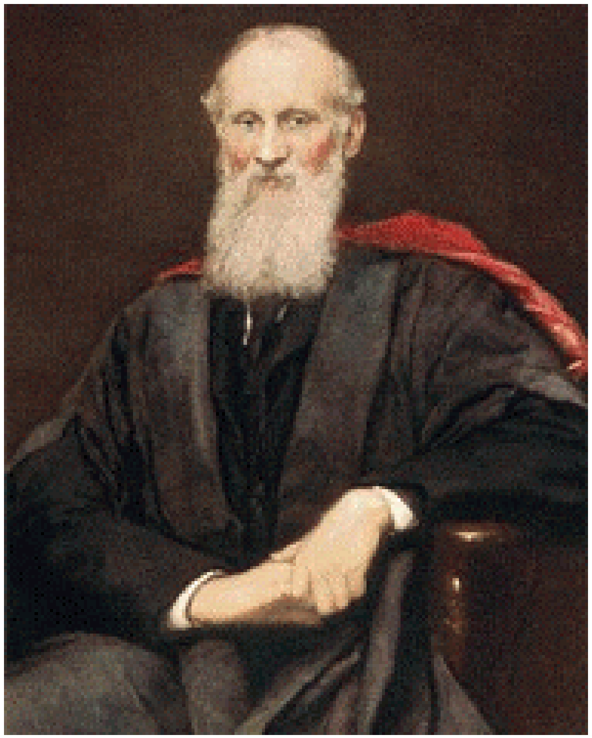
예수님의 신성과 행하신 일들을 믿을 수 있다는 것은 하나님의 선물이지요. 예수님을 믿는 증거란 예수님이 명하신 일들에 순종하는 것입니다.

켈빈(William Thomson 1824~1907)



- 1846~53년 글래스고대학 자연철학 교수
  예감검류계. 상한전위계. 저항측정용 더블브리지.
  전류천칭. 나침반 제작
- 1848년 열역학온도눈금 제안 (단위 K)
  줄-톰슨효과 발견
- 1852년에 열역학 제2법칙 확립 기여

켈빈



“하나님은 나에게 십계명과 더불어 또 한가지 계명을 주셨다. 그 11번째 계명은 이것이다. 과학이 인도하는 곳으로 올라가라. 거기서 지구의 무게를 달고, 공기의 무게도 달며, 조수에 대하여 알아보아라.”

맥스웰(James Clerk Maxwell 1831~1879)

- 1856년 애버딘대학 물리학교수
- 1860년에 런던 킹스칼리지 교수
- 1874년 캐번디시연구소의 초대 소장에 취임
- 1864년 유명한 맥스웰의 기본방정식
- 1873년 ‘전기자기론’ 저술 (전자기파 존재와
  전파 속도가 빛의 속도와 같음을 증명)
- 1879년 앙상블개념을 도입하여
  통계역학 기초 세움

맥스웰

“분자의 유사성을 설명할 수 있는 진화의 가설이란 없다. 왜냐하면 진화론은 끊임없이 변화되어 간다는 것을 항상 전제하고 있기 때문이다.”