연립방정식의 해법과 행렬

고급수치해석
2016. 3. 24

Contents

선형 방정식

- **방정식 각각의 항이 상수와 미지수의 단일 변수의 곱으로 형성**
  - 미지수 n개를 포함하는 선형방정식 : 방정식 n개 필요
  - 주어진 세 개 선형방정식

$$\begin{aligned} 3x_1 + x_2 + 2x_3 &= -4 \\ -x_1 - 2x_2 + x_3 &= -2 \\ 2x_1 + 3x_2 - 2x_3 &= 5 \end{aligned}$$

  - 가감법, 대입법, 등치법 중 하나를 이용 : 변수 x_1, x_2, x_3 계산

$$x_1 = 1, \quad x_2 = -1, \quad x_3 = -3$$

  - 배열 형태로 단순화시킨 선형방정식

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 2 \\ -1 & -2 & 1 \\ 2 & 3 & -2 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} -4 \\ -2 \\ 5 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ -1 \\ -3 \end{bmatrix}$$

3

선형 방정식

- 배열 **A** : 가로줄-행, 세로줄-열
- 행렬 : 가로줄과 세로줄이 여러 개
- 열벡터 : 열이 한 개만 있는 **c**와 **x**
- 행벡터 : 행이 하나인 행렬
- 선형시스템의 표시

$$\mathbf{Ax} = \mathbf{c} \qquad (1)$$

  - ✓ **Ax** : 행렬−벡터의 곱 표시
  - ✓ 비선형시스템을 수치적으로 해석
  - ✓ 응용 수학 부분에서 광범위하게 사용

4

선형 방정식

➢ 배열 형태로 일반화시킨 선형방정식 n개와 변수 n개($x_1, x_2, \cdots, x_n$)

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

➢ $\mathbf{Ax} = \mathbf{c}$로 표시

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= c_1 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= c_n \end{aligned} \qquad (3)$$

✓ 열벡터 $\mathbf{c}$: 식 (1)의 상수만 표시

➢ 동종(homogeneous)의 선형시스템 : 열벡터 $\mathbf{c}$의 모든 원소들이 0인 경우

✓ $\mathbf{Ax} = 0$은 $\mathbf{x} = 0$을 만족하는 $x_1 = x_2 = \cdots = x_n = 0$ 값

행렬 덧셈 연산

- **행렬 표시**

➢ 실수 혹은 복소수를 직사각형 배열로 표시

➢ 행이 m 개, 열이 n 개인 $m \times n$ 행렬 $\mathbf{A}$ 표시

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

➢전치행렬(transposed matrix) : 행렬 $\mathbf{A}$의 행과 열이 서로 바뀐 행렬

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix}$$

✓ 전치행렬의 크기는 $n \times m$

행렬 덧셈 연산

행렬 덧셈 연산

- $m \times n$ 행렬 **A**와 **B**의 덧셈 연산 : 같은 위치에 있는 원소들을 각각 더함
- 연산된 새로운 행렬 **C**의 크기 : $m \times n$

$$\mathbf{C} = \mathbf{A} + \mathbf{B}$$

$$\begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & \cdots & a_{1n}+b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

- 영행렬(zero matrix) : 모든 원소가 **0**인 $m \times n$ 행렬
  - ✓ 표기 방법은 $0_{m\times n}$ 혹은 **0**
  - ✓ 행렬 덧셈의 항등원
- $m \times n$ 행렬 **A**와 **B**를 덧셈하여 얻은 결과 : 영행렬
  - ✓ 행렬 **B**는 행렬 **A**의 모든 원소가 음의 값을 갖는 행렬 −**A**와 동일

행렬 덧셈 연산

예제 1 전치행렬과 덧셈 연산

행렬 **A**와 **B**가 다음과 같다. 전치행렬 $\mathbf{A}^T$와 $\mathbf{B}^T$를 구하고, $\mathbf{A}^T + \mathbf{B}^T$를 구하라

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$$

Tip !

✓ 전치행렬은 주어진 행렬의 행과 열의 순서를 바꿈

행렬 덧셈 연산

MATLAB 1 [예제 1]의 검증

매트랩을 이용하여 [예제 1]의 결과를 확인하라.

Tip !

✓ 매트랩에서 전치행렬은 작은따옴표를 사용

행렬 곱셈 연산

- **행렬 곱셈 연산**
  - ➢ $m \times n$ 행렬 $\mathbf{A}$의 각각의 원소에 임의의 상수 α를 곱함

$$\alpha \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \cdots & \alpha a_{1n} \\ \vdots & \ddots & \vdots \\ \alpha a_{m1} & \cdots & \alpha a_{mn} \end{bmatrix}$$

  - ➢ 행렬 $\mathbf{A}$와 행렬 $\mathbf{B}$의 곱셈 연산
    - ✓ 앞쪽 행렬 $\mathbf{A}$의 열의 수와 뒤쪽 행렬 $\mathbf{B}$의 행의 수가 같은 경우에만 가능
    - ✓ $m \times n$ 행렬 $\mathbf{A}$와 $m \times n$ 행렬 $\mathbf{B}$의 곱셈 연산 $\mathbf{AB}$는 불가능
    - ✓ $m \times n$ 행렬 $\mathbf{A}$와 $n \times p$ 행렬 $\mathbf{B}$의 곱셈 연산 $\mathbf{C} = \mathbf{AB}$는 가능, $\mathbf{C}$의 크기는 $m \times p$

행렬 곱셈 연산

➢ 곱셈 연산의 수식

$$\mathbf{C}_{ij} = \mathbf{A}_{i*}\mathbf{B}_{*j}$$

$$= [a_{i1} \quad a_{i2} \quad \cdots \quad a_{in}]\begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} \qquad (4)$$

$$= a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} \qquad (5)$$

행렬 곱셈 연산

➢ 식 (4)의 계산에서 얻은 선형시스템 수식 $\mathbf{Ax} = \mathbf{c}$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n \end{bmatrix}$$

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= c_1 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= c_n \end{aligned}$$

➢ 정방행렬(square matrix)은 행과 열의 수가 같은 행렬

✓ 크기는 $n \times n$

➢ 단위행렬(identity matrix)은 정방행렬의 주대각선에 위치한 모든 원소가 1

✓ 나머지 원소는 0인 행렬

행렬 곱셈 연산

- $n \times n$ 단위행렬은 $\mathbf{I}_n$으로 표기, 간략하게 $\mathbf{I}$로 표기하고 기본 연산의 1과 같은 개념
- 단위행렬의 일반적 표시

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & 1 \end{bmatrix}$$

- 분할행렬(partitioned matrix)은 양방향 , 수직선, 수평선 방향으로 나누어진 행렬

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & -1 & 5 \end{bmatrix} = \begin{bmatrix} B & c \\ d & e \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0 & -1 \end{bmatrix}, \quad \mathbf{e} = 5$$

- 작은 행렬들이 모여서 구성 혹은 큰 행렬을 여러 개의 작은 행렬로 나누어서 구성

행렬 곱셈 연산

예제 2 행렬 곱셈 연산

행렬 $\mathbf{A}$와 $\mathbf{B}$가 다음과 같다. 곱셈 연산 $\mathbf{AB}$와 $\mathbf{BA}$를 계산하라.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Tip !

- ✓ 곱셈 연산 $\mathbf{AB}$는 새로운 2×2 행렬을 표시
- ✓ 곱셈 연산 $\mathbf{BA}$는 새로운 3×3 행렬을 표시

행렬 곱셈 연산

MATLAB 2 [예제 2]의 검증

매트랩을 이용하여 [예제 2]의 결과를 확인하라.

Tip !

✓ 매트랩을 이용한 행렬의 곱셈은 *(별표)를 이용

역행렬 연산

- **역행렬(inverse matrix)**
  - 행렬방정식 **Ax** = **c**에서 **x**에 대한 해를 구함
    - ✓ 상수 벡터 **c**를 행렬 **A**로 나눔
  - 행렬 **A**가 분모 위치 놓여 $\mathbf{A}^{-1}$ $\left(\frac{\mathbf{c}}{\mathbf{A}}\right)$로 표시
  - 단위행렬을 이용한 행렬과 역행렬의 관계

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \qquad (6)$$

  - **Ax** = **c**의 양변에 $\mathbf{A}^{-1}$을 곱하고 식 (6) 이용

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{A}^{-1}\mathbf{c}, \qquad \mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{I}\mathbf{x} = \mathbf{x}$$

  - **x**에 대한 해

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{c} \qquad (7)$$

역행렬 연산

- **행렬식(determinant)**
  - 행과 열의 수가 같은 정방행렬에 수를 대응시킨 함수
    - ✓ |**A**| 또는 det(**A**)
  - 정칙행렬(nonsingular matrix)은 행렬식의 값이 0이 아닌, | **A** |≠ 0|행렬
  - 행렬식의 값이 0인, |**A**| = 0, 비정칙행렬(singular matrix)은 역행렬 계산 불가능
  - 역행렬 $\mathbf{A}^{-1}$ 이 존재하려면 행렬 **A**는 반드시 정방행렬이고 정칙행렬

역행렬 연산

- **행렬 A의 역행렬 구하기**

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

❶ 정칙행렬 조건을 만족하는 **A**의 행렬식을 구한다.

$$\det(\mathbf{A}) = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{32}a_{21} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{32}a_{23} \neq 0$$

❷ a_{11}이 포함되는 행과 열을 제외한 나머지 원소만으로 2×2 크기의 새로운 행렬에 대한 행렬식을 만들고, M_{11}로 지정한다. 이를 a_{11}에 대한 **소행렬식**(minor)이라고 한다. 같은 방법으로 나머지 원소들에 대한 소행렬식을 구한다.

$$\det(M_{11}) = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} = a_{22}a_{33} - a_{23}a_{32}$$

역행렬 연산

❸ a_{11} 자리에 M_{11} 값을 지정하고 다른 자리에도 소행렬식 방법으로 구한 나머지값을 지정한 뒤 행과 열을 전치시킨다. 이렇게 구한 행렬을 행렬 $\mathbf{A}$에 대한 **수반행렬**(adjoint)이라고 하고, $Adj(\mathbf{A})$로 표시한다.

$$Adj(\mathbf{A}) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}^T = \begin{bmatrix} M_{11} & M_{21} & M_{31} \\ M_{12} & M_{22} & M_{32} \\ M_{13} & M_{23} & M_{33} \end{bmatrix}$$

❹ 수반행렬의 **여인자**(cofactor)를 지정한다. 여인자는 행과 열의 수를 더한 결과가 짝수이면 양의 부호 (+)를 붙이고, 홀수이면 음의 부호 (−)를 지정하는 것을 말한다.

$$\begin{bmatrix} +M_{11} & -M_{21} & +M_{31} \\ -M_{12} & +M_{22} & -M_{32} \\ +M_{13} & -M_{23} & +M_{33} \end{bmatrix}$$

역행렬 연산

❺ 과정 ❹에서 구한 행렬을 과정 ❶에서 구한 $\mathbf{A}$의 행렬식으로 나누면 최종적으로 $\mathbf{A}$의 역행렬 $\mathbf{A}^{-1}$을 얻는다.

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} +M_{11} & -M_{21} & +M_{31} \\ -M_{12} & +M_{22} & -M_{32} \\ +M_{13} & -M_{23} & +M_{33} \end{bmatrix}$$

- 역행렬 $\mathbf{A}^{-1}$을 3행 1열 상수 벡터 $\mathbf{c}$에 곱하면 해 $\mathbf{x}$를 구함
- 크래머(Cramer) 법칙과 매트랩의 왼쪽 나눗셈을 이용해서도 해 $\mathbf{x}$를 구함

역행렬 연산

예제 3 역행렬 계산

다음 행렬의 역행렬을 구하라.

(a) $\mathbf{A} = \begin{bmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{bmatrix}$ (b) $\mathbf{B} = \begin{bmatrix} 1 & 5 & 3 \\ 2 & 8 & 6 \\ 1 & 2 & 3 \end{bmatrix}$

Tip !

✓ 행렬식이 0이 아닌 정칙행렬만 역행렬 연산이 가능

역행렬 연산

MATLAB 3 [예제 3]의 검증

[예제 4-3]에서 다음 행렬의 역행렬을 구했다. 그 결과를 매트랩으로 확인하라.

(a) $\mathbf{A} = \begin{bmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{bmatrix}$ (b) $\mathbf{B} = \begin{bmatrix} 1 & 5 & 3 \\ 2 & 8 & 6 \\ 1 & 2 & 3 \end{bmatrix}$

Tip !

✓ 매트랩에서 행렬 **A**에 대한 행렬식 명령어는 **det(A)**
✓ 매트랩에서 행렬 **A**에 대한 역행렬 명령어는 **inv(A)**

역행렬 연산

예제 4 크래머 법칙

다음과 같은 선형방정식이 있다. 선형방정식의 해를 크래머 법칙을 이용하여 계산하라.

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

Tip !

✓ 배열 형태로 일반화시킨 선형시스템

$$\mathbf{A} = \begin{bmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

크래머법칙 이용 계산

$$\mathbf{A} = \begin{bmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x_1 = \frac{\begin{vmatrix} 4 & 3 & 4 \\ 2 & 1 & 2 \\ 1 & -1 & 1 \end{vmatrix}}{\begin{vmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{vmatrix}} \qquad x_2 = \frac{\begin{vmatrix} -1 & 4 & 4 \\ -1 & 2 & 2 \\ 3 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{vmatrix}} \qquad x_3 = \frac{\begin{vmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{vmatrix}}{\begin{vmatrix} -1 & 3 & 4 \\ -1 & 1 & 2 \\ 3 & -1 & 1 \end{vmatrix}}$$

역행렬 연산

MATLAB 4 **역행렬과 왼쪽 나눗셈**

[예제 4]의 결과를 매트랩의 왼쪽 나눗셈 연산을 이용하여 확인하라.

Tip !

✓ 왼쪽 나눗셈은 컴퓨터 글자판에 따라 ₩ 혹은 \를 사용

기본행 연산

- **기본행(elementary row) 연산**
  - ➢ 기본 행렬(elementary matrix)을 만드는 연산
  - ➢ 어떤 행렬에 기본 행렬을 여러 번 곱하여 $n \times n$ 크기의 단위행렬 계산
    - ✓ 역행렬을 구할 수 있음
  - ➢ 기본행 연산의 과정

❶ 0이 아닌 상수값을 한 행에 곱한다.
❷ 한 행의 0이 아닌 배수를 다른 행에 더한다.
❸ 두 행을 서로 교환한다.
❹ $n \times n$ 크기의 단위행렬 형태가 생성될 때까지 필요한 과정을 반복한다.

기본행 연산

- **3 × 4 행렬 D에 대한 기본행 연산**

$$\mathbf{D}=\begin{bmatrix}-1 & 3 & 4 & 4\\ -1 & 1 & 2 & 2\\ 3 & -1 & 1 & 1\end{bmatrix}$$

➢ 2행에 −2를 곱하여 1행에 더하고, 3행에 −2를 곱하여 2행에 더함

$$\begin{bmatrix}1 & 1 & 0 & 0\\ -7 & 3 & 0 & 0\\ 3 & -1 & 1 & 1\end{bmatrix}$$

✓ 곱하는 상수값, 더하는 행에 대한 뚜렷한 공식은 없음

✓ 과정 ①~④를 이용하여 적절하게 찾음

✓ 가우스 소거법에서 규칙적인 방법을 제시

기본행 연산

➢ 1행에 7를 곱하여 2행에 더하고, 1행에 −3을 곱하여 3행에 더함

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 10 & 0 & 0\\ 0 & -4 & 1 & 1\end{bmatrix}$$

➢ 2행에 $\frac{1}{10}$을 곱함

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & -4 & 1 & 1\end{bmatrix}$$

➢2행에 -1를 곱하여 1행에 더하고, 2행에 4를 곱하여 3행에 더함

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

기본행 연산

예제 5 기본행 연산

행렬 $\mathbf{D}$에 기본행 연산을 실행해보자.

$$\mathbf{D}=\begin{bmatrix}3&3&3&1\\2&2&3&1\\1&2&3&1\end{bmatrix}$$

Tip !

✓ 3×3 크기의 단위행렬이 형성되면 기본행 연산이 완료

기본행 연산

[예제 5] 보충 설명

- 선형연립방정식으로 행렬 $\mathbf{D}$를 표시

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

- 배열 형태로 변환

$$\mathbf{A}=\begin{bmatrix}3&3&3\\2&2&3\\1&2&3\end{bmatrix},\quad \mathbf{c}=\begin{bmatrix}1\\1\\1\end{bmatrix},\quad \mathbf{x}=\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$$

- 행렬 $\mathbf{A}$와 상수 벡터 $\mathbf{c}$를 확장시켜 만든 행렬에 기본행 연산을 실행
- 기본행 연산으로 구한 해

$$x_1=0\,,\;\; x_2=0\,,\;\; x_3=\frac{1}{3}$$

행렬의 나눗셈

- A*B=C라 하면, 나눗셈은 A=C/B 혹은 B=A₩C와 같이 (우측)나눗셈과 (좌측)나눗셈 두 가지로 정의

- $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} / \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix} = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$
  $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} = \begin{bmatrix} x & y \\ z & w \end{bmatrix} \times \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix} = \begin{bmatrix} xg + yi & xh + yk & xi + yl \\ zg + wj & zh + wk & zi + wl \end{bmatrix}$

31

의사 역행렬 (Pseudo inverse)

- 역행렬의 계산은 m=n일 때 가능 (행렬간의 나눗셈 C/D 은, 이 경우 벡터" D " 가 역 행렬을 가져야 함), 매트랩에서 벡터의 나눗셈 연산 '/' 은 ?

  예) A =[1 2 3], B = [4 5 6] 에 대하여
  - ✓ A/ B = 0.4156
- Matlab은 " D "의 역 행렬대신 의사 역 행렬(pseudo inverse)를 구하여 계산
- A의 의사 역행렬 : A^+
  - ✓ $A^+ = (A^T * A)^{-1} * A^T$ (($A^T * A$)의 역행렬이 존재하는 경우)
  - ✓ $A^+ = V\sum^+ U^T$ (특이값 분해(SVD)를 이용하여 계산, $A= U\sum V^T$ 라 할 때)
  - ✓ pinv(A)

✓ 과제 : 특이값 분해 (예를 들어 설명하기)

32

순수 가우스 소거법

- 가우스 소거법(Gauss elimination)

  : 방정식 n 개와 미지수 n 개를 나타내는 선형시스템을 풀 때, n 번째 방정식에 미지수가 한 개(x_n) 남을 때까지 계속 소거하는 방법

- 순수 가우스 소거법
  - ➢ 피봇팅을 사용하지 않는 가우스 소거법
    - ✓ 피봇팅 방법은 행렬의 자리바꿈을 이용하는 가우스 소거법
  - ➢ 순수 가우스 소거법의 전체 과정

❶ **피봇**(pivot) 방정식이라고 부르는 첫 번째 방정식에 적당한 인수를 곱하고, 다른 방정식의 변수 하나를 소거하기 위해서 더하거나 빼준다. 같은 피봇 방정식을 이용하여 나머지 방정식에도 같은 과정을 실행한다. 이러한 과정을 거치면 첫 번째 방정식을 제외한 나머지 방정식들은 변수 하나가 줄어든 새로운 방정식 세트를 형성한다.

33

순수 가우스 소거법

❷ 새롭게 형성된 방정식 세트의 첫 번째 방정식을 다시 새로운 피봇 방정식으로 놓고 앞의 과정을 계속 실행한다. 방정식 하나, 변수 하나가 남을 때까지 이 과정을 반복한다. 이 방법은 컴퓨터로 구현하기 적합하여 컴퓨터로 선형방정식을 풀기 위해 고안된 많은 방법의 기초 형태로 이용된다.

❸ 변환된 상수값 c_n을 변환된 x_n의 계수로 나누면 x_n의 해를 구할 수 있다. 이어서 n번째 방정식에서 계산된 x_n의 값을 바로 앞 $n-1$번째 방정식에 대입하면 x_{n-1}의 해도 얻을 수 있다. 같은 방법으로 계속 **후진 대입법**(backward substitution)을 실행하면 첫 번째 방정식에서 x_1의 해까지 구할 수 있다(후진 대입법은 172p 참조).

34

순수 가우스 소거법

- 식 (2)에 대한 가우스 소거법

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

➢ 배열 **A**와 **c**를 함께 묶은 확대행렬(augmented matrix)

$$[\mathbf{A}^{(1)} \mid \mathbf{c}^{(1)}] = \left[\begin{array}{cccc|c} \underline{a_{11}^{(1)}} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & c_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \ddots & a_{2n}^{(1)} & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & c_n^{(1)} \end{array}\right]$$

✓ 위첨자 (1)은 확대행렬로 변환시킨 후의 초기 원소

✓ 변수를 하나씩 소거할 때마다 위첨자는 (2), (3), …으로 증가

순수 가우스 소거법

➢ 피봇 원소 혹은 선행 계수는 각 행에서 처음 나타나는 0이 아닌 원소

➢ 식 (8)에서 밑줄 친 1행의 원소 $a_{11}^{(1)}$가 피봇 원소

✓ 2행부터 n 번째 행까지 변수 x_1을 소거할 때 사용

➢ 승수(multiplier) 구하기

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} \qquad (i = 2, \ \cdots, \ n)$$

✓ 피봇 원소와 나머지 각 행의 첫 번째 원소의 곱셈 비례

➢ 변수 x_1을 차례로 소거시키고 계산된 새로 지정되는 행렬 원소

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} \qquad (j = 2, \ \cdots, \ n)$$

$$c_i^{(2)} = c_i^{(1)} - m_{i1}c_1^{(1)}$$

➢ 열을 표시하는 지표 j가 1을 포함하지 않는 이유

$$a_{i1}^{(2)} = a_{i1}^{(1)} - m_{i1}a_{11}^{(1)} = 0$$

순수 가우스 소거법

- 새로운 확대행렬

$$[\mathbf{A}^{(2)} \mid \mathbf{c}^{(2)}] = \left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & c_1^{(1)} \\ 0 & \underline{a_{22}^{(2)}} & \ddots & a_{2n}^{(2)} & c_2^{(2)} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & c_n^{(2)} \end{array}\right]$$

- 식 (9)에서 밑줄 친 2행의 원소 가 $a_{22}^{(2)}$봇 원소
  - ✓ 3행부터 n 번째 행까지 변수 x_2를 소거할 때 사용
- 승수(multiplier) 구하기

$$m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} \qquad (i = 3, \ \cdots, \ n)$$

순수 가우스 소거법

- 변수 x_2를 차례로 소거시키고 계산된 새로 지정되는 행렬 원소

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2} a_{2j}^{(2)} \qquad (i = 3, \ \cdots, \ n)$$

$$c_i^{(3)} = c_i^{(2)} - m_{i2} c_2^{(2)}$$

- n 번째 행에 변수 x_n만 남을 때까지 계속 반복하고 생성된 확대행렬

$$[\mathbf{A}^{(n)} \mid \mathbf{c}^{(n)}] = \left[\begin{array}{cccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & \cdots & a_{1n}^{(n)} & c_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} & c_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & \cdots & a_{3n}^{(3)} & c_3^{(3)} \\ \vdots & \cdots & 0 & \ddots & \cdots & \vdots & \vdots \\ \vdots & \cdots & \cdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{nn}^{(n)} & c_n^{(n)} \end{array}\right] \text{0)}$$

순수 가우스 소거법

- n 번째 행에서 구한 변수 x_n

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

- $n-1$ 번째 행에 변수 x_{n-1}의 값을 얻음
- 가우스 소거법의 후진 대입법을 이용
  - ✓ 역순으로 최종적으로 1행에서 x_1의 값 얻음

39

순수 가우스 소거법

- **세 개의 선형방정식에서 x의 해 구하기**

$$\begin{aligned} -x_1 + x_2 + 2x_3 &= 2 \qquad (E1) \\ 3x_1 - x_2 + x_3 &= 6 \qquad (E2) \\ -x_1 + 3x_2 + 4x_3 &= 4 \qquad (E3) \end{aligned}$$

- **(1) 수학적 비법**
  - 방정식 $(E1)$이 피봇 방정식
  - $(E1)$에 3을 곱하고 그 결과를 $(E2)$에 더함
  - $(E1)$에 -1을 곱하고 그 결과를 $(E3)$에 더함
  - x_1이 소거된 새로운 방정식

$$\begin{aligned} -x_1 + x_2 + 2x_3 &= 2 \qquad (E1) \\ 2x_2 + 7x_3 &= 12 \qquad (E2) \\ 2x_2 + 2x_3 &= 2 \qquad (E3) \end{aligned}$$

40

순수 가우스 소거법

- 방정식 ($E2$)가 새로운 피봇 방정식
- ($E2$)에 −1을 곱하고 그 결과를 ($E3$)에 더함
- x_2가 소거된 새로운 방정식

$$\begin{aligned} -x_1 + x_2 + 2x_3 &= 2 & (E1) \\ 2x_2 + 7x_3 &= 12 & (E2) \\ -5x_3 &= -10 & (E3) \end{aligned}$$

- ($E3$)에 x_3를 풀면 2
- 후진 대입법을 이용하여 변수들을 찾음
  - ✓ $x_2 = -1,\ x_1 = 1$

순수 가우스 소거법

- **(2) 가우스 소거법**
  - 배열 형태로 변화시킨 선형방정식

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 2 \\ 3 & -1 & 1 \\ -1 & 3 & 4 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 2 \\ 6 \\ 4 \end{bmatrix}$$

  - 배열 **A**와 **c**를 함께 묶은 확대행렬

$$[\mathbf{A}^{(1)} \mid \mathbf{c}^{(1)}] = \left[\begin{array}{ccc|c} -1 & 1 & 2 & 2 \\ 3 & -1 & 1 & 6 \\ -1 & 3 & 4 & 4 \end{array}\right] \begin{array}{c} (R1) \\ (R2) \\ (R3) \end{array}$$

  - 1행 1열의 −1을 피봇 원소로 이용하여 구한 2행과 3행의 승수

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = \frac{3}{-1} = -3 \qquad m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = \frac{-1}{-1} = 1$$

순수 가우스 소거법

➢ 각 행의 변수 x_1을 소거하고 새로 지정된 행렬 원소들의 계산

$$a_{22}^{(2)} = a_{22}^{(1)} - m_{21}a_{12}^{(1)} = -1-(-3)(1) = 2$$

$$a_{23}^{(2)} = a_{23}^{(1)} - m_{21}a_{13}^{(1)} = 1-(-3)(2) = 7$$

$$a_{32}^{(2)} = a_{32}^{(1)} - m_{31}a_{12}^{(1)} = 3-(1)(1) = 2$$

$$a_{33}^{(2)} = a_{33}^{(1)} - m_{31}a_{13}^{(1)} = 4-(1)(2) = 2$$

$$c_2^{(2)} = c_2^{(1)} - m_{21}c_1^{(1)} = 6-(-3)(2) = 12$$

$$c_3^{(2)} = c_3^{(1)} - m_{31}c_1^{(1)} = 4-(1)(2) = 2$$

$$[\mathbf{A}^{(2)} \mid \mathbf{c}^{(2)}] = \left[\begin{array}{ccc|c} -1 & 1 & 2 & 2 \\ 0 & 2 & 7 & 12 \\ 0 & 2 & 2 & 2 \end{array}\right]$$

43

순수 가우스 소거법

➢ 2행 2열의 2를 피봇 원소로 이용하여 구한 3행의 승수

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = \frac{2}{2} = 1$$

➢ 각 행의 변수 x_2를 소거하고 새로 지정된 행렬 원소들의 계산

$$a_{33}^{(3)} = a_{33}^{(2)} - m_{32}a_{23}^{(2)} = 2-(1)(7) = -5$$

$$c_3^{(3)} = c_3^{(2)} - m_{32}c_2^{(2)} = 2-(1)(12) = -10$$

$$[\mathbf{A}^{(3)} \mid \mathbf{c}^{(3)}] = \left[\begin{array}{ccc|c} -1 & 1 & 2 & 2 \\ 0 & 2 & 7 & 12 \\ 0 & 0 & -5 & -10 \end{array}\right]$$

44

순수 가우스 소거법

- 3행에서 변수 x_3

$$x_3 = \frac{c_3^{(3)}}{a_{33}^{(3)}} = \frac{-10}{-5} = 2$$

- 후진 대입법을 이용하여 변수들을 찾음
  - ✓ $x_2 = -1,\ x_1 = 1$

순수 가우스 소거법

예제 6 순수 가우스 소거법

선형시스템 $\mathbf{Ax} = \mathbf{c}$에 대한 행렬이 다음과 같다. 순수 가우스 소거법을 이용하여 변수 $x_1,\ x_2,\ x_3$를 찾아라.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 2 & 2 & 3 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

Tip !

- ✓ 확대 행렬을 구하고 소거법 시작
- ✓ 피봇 원소를 이용하여 승수를 구함

순수 가우스 소거법

MATLAB 5 매트랩을 이용한 가우스 소거법

매트랩을 이용하여 다음 선형시스템의 변수 x_1, x_2, x_3를 구하라.

$$x_1 - x_2 + x_3 = 0$$
$$x_2 - x_3 = 1$$
$$2x_1 + 2x_2 - x_3 = 6$$

피봇팅을 사용한 가우스 소거법

- **피봇팅(pivoting)을 사용한 가우스 소거법**
  - ➢ 피봇팅 방법과 부분 피봇팅 방법으로 구분
  - ➢주대각선에 0이 놓여 있는 경우 행과 열을 단순히 자리바꿈(interchanging)
  - ➢ 0을 주대각선상에서 제거하고 순수 가우스 소거법을 실행
  - ➢ 주대각선상의 원소 0을 제거하지 않은 경우
    - ✓ 피봇 원소를 계속 사용하면 승수를 실행할 때 0으로 나누는 오류 발생
- **부분 피봇팅 방법(partial pivoting)**
  - ➢ 자리바꿈을 허용하는 점은 피봇팅 방법과 동일
  - ➢ 행렬의 1열부터 차례대로 절댓값이 가장 큰 원소를 피봇 원소로 지정
  - ➢ 매번 가장 큰 절댓값 원소를 포함하는 열을 상위 행으로 자리바꿈
    - ✓ 전체적인 행렬변수의 차수에는 영향이 없음

피봇팅을 사용한 가우스 소거법

- **부분 피봇팅을 사용하여 변수 x_1, x_2, x_3 찾기**

$$\begin{bmatrix} -1 & 1 & 2 \\ 1 & 2 & -1 \\ -2 & -2 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \\ 0 \end{bmatrix}$$

➢ 확대행렬 표시

$$\left[\begin{array}{ccc|c} -1 & 1 & 2 & 7 \\ 1 & 2 & -1 & 2 \\ -2 & -2 & 2 & 0 \end{array}\right]$$

➢ 1행과 3행을 자리바꿈

$$\left[\begin{array}{ccc|c} -2 & -2 & 2 & 0 \\ 1 & 2 & -1 & 2 \\ -1 & 1 & 2 & 7 \end{array}\right] \qquad (R1) \leftrightarrow (R3)$$

✓ 가장 큰 절댓값을 피봇 원소로 사용

피봇팅을 사용한 가우스 소거법

➢ 1행 1열의 −2를 피봇 원소로 이용하여 구한 2행과 3행의 승수

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = \frac{1}{-2} = -\frac{1}{2}$$

$$m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = \frac{-1}{-2} = \frac{1}{2}$$

➢ 1행에 $-\frac{1}{2}$을 곱하고 그 결과를 2행에서 뺀 뒤에 2행에 놓음

➢ 1행에 $\frac{1}{2}$을 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\left[\begin{array}{ccc|c} -2 & -2 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 2 & 1 & 7 \end{array}\right] \begin{array}{l} \\ \leftarrow (R2) - \left(-\frac{1}{2}\right) \times (R1) \\ \leftarrow (R3) - \left(\frac{1}{2}\right) \times (R1) \end{array}$$

피봇팅을 사용한 가우스 소거법

➢ 2행과 3행을 자리바꿈하여 큰 절대값을 피봇 원소로 사용

$$\left[\begin{array}{ccc|c} -2 & -2 & 2 & 0 \\ 0 & 2 & 1 & 7 \\ 0 & 1 & 0 & 2 \end{array}\right] \qquad (R2) \leftrightarrow (R3)$$

➢ 2행 2열의 2를 피봇 원소로 이용하여 구한 3행의 승수

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = \frac{1}{2}$$

➢ 2행에 $\frac{1}{2}$를 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\left[\begin{array}{ccc|c} -2 & -2 & 2 & 0 \\ 0 & 2 & 1 & 7 \\ 0 & 0 & -\frac{1}{2} & -\frac{3}{2} \end{array}\right] \quad \leftarrow \quad (R3) - \left(\frac{1}{2}\right) \times (R2)$$

➢ 후진 대입법을 이용하여 구한 변수 $x_3 = 3,\ x_2 = 2,\ x_1 = 1$

피봇팅을 사용한 가우스 소거법

예제 7 **부분 피봇팅 가우스 소거법**

[매트랩 5]에서 제시된 다음 선형시스템의 변수 x_1, x_2, x_3를 부분 피봇팅을 사용한 가우스 소거법을 이용하여 구하라.

$$\begin{aligned} x_1 - x_2 + x_3 &= 0 \\ x_2 - x_3 &= 1 \\ 2x_1 + 2x_2 - x_3 &= 6 \end{aligned}$$

Tip !

✓ 확대 행렬을 구하고 소거법 시작

피봇팅을 사용하지 않는 가우스 조르단 소거법

- **가우스 조르단 소거법(Gauss-Jordan elimination)**

  : 가우스 소거법을 이용하여 역행렬을 구하는 방법

  - 기본행 연산을 이용하여 축약된(reduced) 행 사다리꼴을 얻는 알고리즘
    - ✓ 후진 대입법 과정 불필요

- **행 사다리꼴(row echelon form)**
  - 0이 아닌 원소가 적어도 하나 이상인 행들이 모인 형태
  - 0이 많은 행보다 0이 적은 행들이 항상 위의 자리에 위치
    - ✓ 원소가 모두 0인 행은 행렬의 맨 마지막 행에 위치
  - 원소가 모두 0이 아닌 행에서 첫 번째 0이 아닌 원소를 피봇이라 부름
    - ✓ 혹은 선두계수(leading coefficient)
  - 피봇 원소를 사용하여 소거법을 실행하면 같은 열의 원소들은 모두 0

피봇팅을 사용하지 않는 가우스 조르단 소거법

- 행 사다리꼴의 예

$$\begin{bmatrix} 1 & a_{12} & a_{13} \\ 0 & -2 & a_{23} \\ 0 & 0 & 3 \end{bmatrix}$$

- 축약된 행 사다리꼴은 주대각선의 원소가 모두 1이 되는 대각행렬
  - ✓ 같은 열에서 1을 포함한 행을 제외한 나머지 행의 원소들은 모두 0
- 축약된 행 사다리꼴의 예

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

피봇팅을 사용하지 않는 가우스 조르단 소거법

➢ 가우스 조르단 소거법에서 확대행렬 지정

$[\mathbf{A}_{n\times n}|\mathbf{I}_n]$ 11)

➢ 왼쪽 부분에 $n\times n$ 계수행렬 $\mathbf{A}$를 쓰고, 오른쪽 부분에 $n\times n$ 단위행렬 $\mathbf{I}$ 지정

➢ 왼쪽 부분이 단위행렬의 형태를 가질 때까지 계속 실행

$[\mathbf{I}_n^{-1} \mid \mathbf{A}_{n\times n}^{-1}]$ 12)

✓ 왼쪽에 있는 단위행렬의 역행렬은 단위행렬 자신

✓ 식 (12)의 오른쪽이 구하려는 행렬 $\mathbf{A}$의 역행렬 $\mathbf{A}^{-1}$

➢ 3행 3열 계수행렬 $\mathbf{A}$와 3행 3열 단위행렬 $\mathbf{I}$를 묶은 확대행렬

$$[\mathbf{A}\mid\mathbf{I}] = \left[\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{array}\right]$$

55

피봇팅을 사용하지 않는 가우스 조르단 소거법

정방행렬 A에 대한 역행렬 구하기

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 2 & 2 & 3 \end{bmatrix}$$

➢ 확대행렬

$$[\mathbf{A}\mid\mathbf{I}] = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 2 & 2 & 3 & 0 & 0 & 1 \end{array}\right]$$

➢ 1행 1열의 1을 피봇 원소로 이용하여 구한 2행과 3행의 승수

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = \frac{0}{1} = 0$$

$$m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = \frac{2}{1} = 2$$

56

피봇팅을 사용하지 않는 가우스 조르단 소거법

➢ 1행에 2를 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 2 & 1 & -2 & 0 & 1 \end{array}\right] \quad \leftarrow \quad (R3)-(2)\times(R1)$$

➢ 2행 2열의 1를 피봇 원소로 이용하여 구한 승수

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = \frac{2}{1} = 2$$

➢ 2행에 2를 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 3 & -2 & -2 & 1 \end{array}\right] \quad \leftarrow \quad (R3)-(2)\times(R2)$$

피봇팅을 사용하지 않는 가우스 조르단 소거법

➢ 3행 3열의 3를 피봇 원소로 이용하여 구한 승수

$$m_{13} = \frac{a_{13}^{(3)}}{a_{33}^{(3)}} = \frac{1}{3}$$

➢3행에 $\frac{1}{3}$을 곱하고 그 결과를 1행에서 뺀 뒤에 1행에 놓음

➢ 3행에 $-\frac{1}{3}$을 곱하고 그 결과를 2행에서 뺀 뒤에 2행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{5}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & 1 & 0 & -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 3 & -2 & -2 & 1 \end{array}\right] \quad \begin{array}{l} \leftarrow \quad (R1)-\left(\frac{1}{3}\right)\times(R3) \\ \leftarrow \quad (R2)-\left(-\frac{1}{3}\right)\times(R3) \\ \end{array}$$

피봇팅을 사용하지 않는 가우스 조르단 소거법

➢ 3행 $\frac{1}{3}$을 곱하고 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{5}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & 1 & 0 & -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{array}\right] \leftarrow \left(\frac{1}{3}\right) \times (R3)$$

➢ 행렬 **A**의 역행렬로 전환된 오른쪽 부분

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{5}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

부분 피봇팅을 사용한 가우스 조르단 소거법

- **끝수처리로 인한 오류가 잘 발생하지 않아 상당히 안정적인 방법**
  ➢ 피봇팅을 사용하는 경우에는 부분 피봇팅을 사용하여 역행렬을 구함
- **계수행렬 A에 부분 피봇팅을 사용한 가우스 조르단 소거법**
  ➢ 확대행렬

$$[\mathbf{A} \mid \mathbf{I}] = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 2 & 2 & 3 & 0 & 0 & 1 \end{array}\right]$$

➢큰 절댓값을 피봇 원소로 사용하기 위해서 1행과 3행을 자리바꿈

$$\left[\begin{array}{ccc|ccc} 2 & 2 & 3 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \quad (R1) \leftrightarrow (R3)$$

부분 피봇팅을 사용한 가우스 조르단 소거법

- 1행에 $\frac{1}{2}$을 곱하고 다시 1행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 1 & \frac{3}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \begin{array}{l} \leftarrow \left(\frac{1}{2}\right)\times(R1) \\ \\ \\ \end{array}$$

- 1행에서 3행을 빼고 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 1 & \frac{3}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & \frac{1}{2} & -1 & 0 & \frac{1}{2} \end{array}\right] \begin{array}{l} \\ \\ \leftarrow (R1)-(R3) \end{array}$$

부분 피봇팅을 사용한 가우스 조르단 소거법

- 1행에서 3행을 빼고 1행에 놓음
- 2행에서 3행을 빼고 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & -\frac{3}{2} & 1 & 1 & -\frac{1}{2} \end{array}\right] \begin{array}{l} \leftarrow (R1)-(R3) \\ \\ \leftarrow (R2)-(R3) \end{array}$$

- 마지막 주대각선 위치인 3행 3열을 원소 1인 값으로 변형
- $-\frac{3}{2}$의 역수인 $-\frac{2}{3}$ 3행에 곱한 뒤에 다시 3행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{array}\right] \begin{array}{l} \\ \\ \leftarrow \left(-\frac{2}{3}\right)\times(R3) \end{array}$$

부분 피봇팅을 사용한 가우스 조르단 소거법

- 1행에서 3행을 빼고 1행에 놓음
- 2행에서 3행을 더하고 2행에 놓음

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{5}{3} & \frac{2}{3} & -\frac{1}{3} \\ 0 & 1 & 0 & -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{array}\right] \begin{array}{l} \leftarrow \ (R1)-(R3) \\ \leftarrow \ (R2)+(R3) \\ \end{array}$$

- 행렬 **A**의 역행렬로 전환된 오른쪽 부분

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{5}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{2}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

부분 피봇팅을 사용한 가우스 조르단 소거법

예제 8 가우스 조르단 소거법

두가지 방식의 가우스 조르단 소거법으로 주어진 행렬 A의 역행렬을 찾아라.

(a) 피봇팅을 사용하지 않는 경우

(b) 부분 피봇팅을 사용하는 경우

$$\mathbf{A} = \begin{bmatrix} 0.001 & 10.00 \\ 10.00 & 10.00 \end{bmatrix}$$

Tip !

✓ 다음과 같은 확대행렬 이용

$$[\mathbf{A}\,|\,\mathbf{I}] = \left[\begin{array}{cc|cc} 0.001 & 10.00 & 1 & 0 \\ 10.00 & 10.00 & 0 & 1 \end{array}\right]$$

다양한 상하행렬 분해

- **상하행렬 분해(LU : Lower-Upper)**

  : 가우스 소거법을 기초로 이용하여 선형시스템을 간편하고 안정적으로 푼다.

  ➢ 행렬의 차수가 커지면 발생하는 가우스 소거법의 단점
  - ✓ 연산 카운터가 매우 크게 증가
  - ✓ 컴퓨터의 실행 속도가 감소

  ➢ 상하행렬 분해는 실행 속도가 빠름
  - ✓ 가우스 소거법의 단점 보완

  ➢ 선형시스템의 정방행렬 **A**를 다음과 같이 표시

$$\mathbf{A} = \mathbf{LU} \qquad (13)$$

다양한 상하행렬 분해

➢ **L**은 하삼각행렬(lower triangular matrix) 표시
- ✓ 주대각선을 기준으로 위쪽에 위치하는 모든 원소들이 0인 행렬
- ✓ 순수 가우스 소거법에서 사용한 승수들과 동일

➢U는 상삼각행렬(upper triangular matrix)
- ✓ 주대각선을 기준으로 아래쪽에 위치하는 모든 원소들이 0인 행렬
- ✓ 순수 가우스 소거법에서 마지막 단계의 계수행렬

➢ $n \times n$ 정방행렬 **A**에 대한 상하행렬 분해

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2n} \\ \vdots & \cdots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & \ddots & 0 & \vdots \\ \vdots & \cdots & \cdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & \cdots & u_{2n} \\ 0 & 0 & \ddots & \cdots & \vdots \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & u_{nn} \end{bmatrix}$$

다양한 상하행렬 분해

- 행렬 **L**의 주대각선상 원소는 모두 1로 놓음
  - ✓ 주대각선 위쪽의 모든 원소는 0으로 놓음
- 행렬 **A**의 주대각선 아래쪽 원소들이 0
  - ✓ 같은 자리의 행렬 **L**의 원소도 0으로 대치
- 행렬 **U**는 주대각선상과 주대각선 위쪽의 모든 원소는 행렬 **U**의 원소로 놓음
  - ✓ 주대각선 아래쪽의 모든 원소는 0으로 놓음
- 행렬 **A**의 주대각선 위쪽의 원소들이 0
  - ✓ 같은 자리의 행렬 **U**의 원소도 0으로 대치
- 식 (13)을 이용하여 선형시스템을 다시 쓰기 ($\mathbf{A} = \mathbf{LU}$

$$\mathbf{LUx} = \mathbf{c} \qquad (15)$$

다양한 상하행렬 분해

- 두 단계로 나누어 쓴 식 (15)

$$\mathbf{Ly} = \mathbf{c} \qquad (16)$$
$$\mathbf{Ux} = \mathbf{y}$$

- 시스템 $\mathbf{Ly} = \mathbf{c}$는 하삼각 행렬 **L**을 포함하는 선형시스템 행태

$$\begin{aligned}
y_1 &= c_1 \quad \cdots\cdots ❶\\
l_{21}y_1 + y_2 &= c_2 \quad \cdots\cdots ❷\\
l_{31}y_1 + l_{32}y_2 + y_3 &= c_3 \quad \cdots\cdots ❸ \; 7)\\
\vdots \quad &\; \vdots \quad \vdots\\
l_{n1}y_1 + \cdots + l_{n,n-1}y_{n-1} + y_n &= c_n \quad \cdots\cdots ⓝ
\end{aligned}$$

- 전진대입법(forward substitution)으로 풀어서 변수 벡터 **y**를 구함
- ①번을 이용하여 미지수 y_1을 구하고, ②번에 y_1을 대입하여 y_2를 구함
- 마지막 ⓝ번에서 미지수 y_n의 값을 구함
- 시스템 $\mathbf{Ux} = \mathbf{y}$에 후진 대입법을 이용하여 변수 벡터 **x**를 최종적으로 구함

다양한 상하행렬 분해

- **선형시스템의 미지의 변수 벡터 구하기**

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

- **방법 1**
  - 가우스 소거법과 거의 같은 과정
  - 1행 1열의 −2를 피봇 원소로 이용하여 구한 2행과 3행의 승수

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{1}{2}$$

$$m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = 0$$

다양한 상하행렬 분해

- 1행 $-\frac{1}{2}$을 곱하고 그 결과를2행에서 뺀 뒤에 2행에 놓음

$$\begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & -1 & 2 \end{bmatrix} \quad \leftarrow \quad (R2) - \left(-\frac{1}{2}\right) \times (R1)$$

- 2행의 $\frac{1}{2}$을 피봇 원소로 이용하여 구한 3행의 승수

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = \frac{-1}{\frac{1}{2}} = -2$$

- 2행에 −2를 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 4 \end{bmatrix} \quad \leftarrow \quad (R3) - (-2) \times (R2)$$

다양한 상하행렬 분해

- 3×3 행렬에 대한 하삼각행렬 **L**과 상삼각행렬 **U**

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

- 실제 구한 원소들의 행렬

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 4 \end{bmatrix}$$

- $\mathbf{Ly} = \mathbf{c}$의 식에 전진 대입법을 이용하여 변수 벡터 $\mathbf{y}$를 구하는 식

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

다양한 상하행렬 분해

- 구한 변수 벡터 $\mathbf{y}$의 값

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}$$

- $\mathbf{Ux} = \mathbf{y}$의 식에 후진 대입법을 이용하여 변수 벡터 $\mathbf{x}$를 구하는 식

$$\begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}$$

- 상하행렬 분해를 이용하여 구한 변수 벡터 $\mathbf{x}$의 최종 결과

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{7}{2} \\ \frac{9}{4} \end{bmatrix}$$

다양한 상하행렬 분해

- **방법 2**
  - ➢ 행렬 **L**과 행렬 **U**의 곱셈 연산을 실행한 뒤에 행렬 **A**의 원소와 비례 관계 이용
  - ➢ 식 (14)의 행렬 곱셈 연산

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ l_{21}u_{11} & l_{21}u_{12}+u_{22} & l_{21}u_{13}+u_{23} \\ l_{31}u_{11} & l_{31}u_{12}+l_{32}u_{22} & l_{31}u_{13}+l_{32}u_{23}+u_{33} \end{bmatrix}$$

  - ➢ 왼쪽 행렬과 오른쪽 행렬의 원소를 각각 비교
    - ✓ 하삼각행렬 **L**과 상삼각행렬 **U**에서 사용할 원소 구함
  - ➢ 필요한 과정과 알고리즘(algorithm)으로 구현하는 방법을 고찰
  - ➢ 알고리즘은 문제 풀이를 위한 절차나 방법을 쉽게 단계적인 순서로 기술
    - ✓ 컴퓨터 프로그램에서는 컴퓨터가 수행할 동작의 순서로 이용

73

다양한 상하행렬 분해

- ➢ 행렬 **A** 1행의 원소는 행렬 **U** 1행의 원소와 같음

$$u_{1i} = a_{1i} \qquad \text{(A1)}$$

  - ✓ 첨자 *i*는 1, 2, 3을 순차적으로 이용
- ➢ 행렬 **U**의 1행

$$u_{11} = a_{11} = -2$$

$$u_{12} = a_{12} = 1$$

$$u_{13} = a_{13} = 0$$

- ➢ 나머지 원소를 구하기 위한 알고리즘

$$l_{i,i-1} = \frac{a_{i,i-1}}{u_{i-1,i-1}} \qquad \text{A2)}$$

$$u_{ii} = a_{ii} - l_{i,i-1}u_{i-1,i} \qquad \text{A3)}$$

  - ✓ 첨자 *i*는 2와 3을 순차적으로 이용

74

다양한 상하행렬 분해

- 식 (9), 식 (A2), 식 (A3)이용하여 나머지 원소를 구하는 과정

$$l_{21} = \frac{a_{21}}{u_{11}} = -\frac{1}{2}$$

$$u_{22} = a_{22} - l_{21}u_{12} = 0 - \left(-\frac{1}{2}\right)(1) = \frac{1}{2}$$

$$u_{23} = a_{23} - l_{21}u_{13} = 1 - \left(-\frac{1}{2}\right)(0) = 1$$

$$l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} = \frac{-1-0}{1/2} = -2$$

$$u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} = 2 - 0 - (-2)(1) = 4$$

- 행렬 **L**과 **U**를 다시 구성

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 4 \end{bmatrix}$$

75

다양한 상하행렬 분해

- $\mathbf{Ly} = \mathbf{c}$의 형태

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

- 전진 대입법을 이용하여 구하는 첫 번째 변수 벡터 $\mathbf{y}$

$$y_1 = c_1 = 2$$

- 나머지 변수들을 구하기 위한 알고리즘, 첨자 i는 2와 3을 순차적으로 이용

$$y_i = c_i - l_{i,i-1}y_{i-1} \quad \text{A4)}$$

- 식 (A4)를 이용하여 변수를 구하는 과정

$$y_2 = c_2 - l_{21}y_1 = 3 - \left(-\frac{1}{2}\right)(2) = 4$$

$$y_3 = c_3 - l_{32}y_2 = 1 - (-2)(4) = 9$$

76

다양한 상하행렬 분해

➢ 변수 벡터 **y** 값

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}$$

➢ **y** 값을 적용한 **Ux** = **y**의 형태

$$\begin{bmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}$$

➢ 첫 번째 변수 벡터 **x**를 구하기 위한 알고리즘

$$x_n = \frac{y_n}{u_{nn}} \qquad \text{(A5)}$$

✓n은 행렬의 가장 큰 차수를 표시

다양한 상하행렬 분해

➢ 식 (A5) 이용

$$x_3 = \frac{y_3}{u_{33}} = \frac{9}{4}$$

➢ 후진 대입법을 이용하여 나머지 변수들을 구하는 알고리즘

$$x_i = \frac{y_i - u_{i,i+1}x_{i+1}}{u_{ii}} \qquad \text{(A6)}$$

✓ 첨자 i는 2와 1을 순차적으로 이용

➢ 식 (A6)을 이용하여 나머지 변수들을 구하는 과정

$$x_2 = \frac{y_2 - u_{23}x_3}{u_{22}} = \frac{4-(1)(9/4)}{1/2} = \frac{7}{2}$$

$$x_1 = \frac{y_1 - u_{12}x_2}{u_{11}} = \frac{2-(1)(7/2)}{-2} = \frac{3}{4}$$

다양한 상하행렬 분해

- 상하행렬 분해를 이용하여 구하는 변수 벡터 **x**

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{7}{2} \\ \frac{9}{4} \end{bmatrix}$$

- 제시한 알고리즘들은 일반화시킨 형태는 아님
  - ✓ 주어진 선형시스템에 알맞게 적용 시킨 형태

다양한 상하행렬 분해

- **방법 3**
  - 방법 2를 변형 사용
  - 행렬 **L**의 주대각선과 주대각선 아래쪽의 모든 원소는 행렬 **L**의 원소를 놓음
    - ✓ 주대각선 위쪽의 원소는 전부 0으로 놓음
  - 행렬 **U**의 주대각선 원소는 전부 1로 놓음
    - ✓ 주대각선 위쪽의 모든 원소는 행렬 **U**의 원소
    - ✓ 주대각선 아래쪽의 원소는 전부 0으로 놓음
  - 행렬 **A**에서 주대각선 위쪽과 아래쪽의 원소들 중 이미 0이 존재
    - ✓ 행렬 **L**과 행렬 **U**의 같은 자리의 원소들도 0으로 바꾸어 사용
  - 행렬 **A**에서 1행 3열과 3행 1열의 값은 0
    - ✓ 행렬 **U**의 u_{13}과 행렬 **L**의 l_{31}도 0으로 놓음

다양한 상하행렬 분해

- 변형시킨 행렬 **L**과 **U**

$$\mathbf{L}=\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ 0 & l_{32} & l_{33} \end{bmatrix}, \quad \mathbf{U}=\begin{bmatrix} 1 & u_{12} & 0 \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

- **A** = **LU** 형태의 연산

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

- 하삼각행렬 **L**과 상삼각행렬 **U**에서 사용할 원소

$$l_{11}=a_{11}=-2$$

- 주대각선 위쪽 행렬 **U**의 원소들을 구하는 알고리즘

$$u_{i,i+1}=\frac{a_{i,i+1}}{l_{ii}} \qquad \text{(A7)}$$

다양한 상하행렬 분해

- 첨자 i에 1을 이용하여 구하는 행렬

$$u_{12}=\frac{a_{12}}{l_{11}}=-\frac{1}{2}$$

- 주대각선 아래 행렬 **L**의 원소들을 구하는 알고리즘

$$l_{i,i-1}=a_{i,i-1} \qquad \text{(A8)}$$

- 첨자 i에 2를 이용하여 구하는 행렬

$$l_{21}=a_{21}=1$$

- 주대각선에 있는 행렬 **L**의 원소들을 구하는 알고리즘

$$l_{ii}=a_{ii}-l_{i,i-1}u_{i-1,i} \qquad \text{(A9)}$$

- 첨자 i에 2를 이용하여 구하는 행렬

$$l_{22}=a_{22}-l_{21}u_{12}=0-(1)\left(-\frac{1}{2}\right)=\frac{1}{2}$$

다양한 상하행렬 분해

- (A7)에서 첨자 $i = 2$를 이용하여 구하는 행렬

$$u_{23} = \frac{a_{23}}{l_{22}} = \frac{1}{1/2} = 2$$

- (A8)에서 첨자 $i = 3$을 이용하여 구하는 행렬

$$l_{32} = a_{32} = -1$$

- (A9)에서 첨자 $i = 3$을 이용하여 구하는 행렬

$$l_{33} = a_{33} - l_{32}u_{23} = 2 - (-1)(2) = 4$$

- 계산된 각각의 원소를 이용하여 나타내는 행렬 $\mathbf{L}$과 $\mathbf{U}$

$$\mathbf{L} = \begin{bmatrix} -2 & 0 & 0 \\ 1 & \frac{1}{2} & 0 \\ 0 & -1 & 4 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

다양한 상하행렬 분해

- $\mathbf{Ly} = \mathbf{c}$의 형태

$$\begin{bmatrix} -2 & 0 & 0 \\ 1 & \frac{1}{2} & 0 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

- 전진 대입법을 이용하여 구하는 첫 번째 변수 벡터 $\mathbf{y}$

$$y_1 = \frac{c_1}{l_{11}} = \frac{2}{-2} = -1$$

- 나머지 변수들을 구하는 알고리즘

$$y_i = \frac{1}{l_{ii}}[c_i - l_{i,i-1}y_{i-1}] \qquad \text{(A10)}$$

  - ✓ 첨자 i는 2와 3을 순차적으로 이용

다양한 상하행렬 분해

- 식 (A10)을 이용하여 변수들을 구하는 과정

$$y_2 = \frac{1}{l_{22}}[c_2 - l_{21}y_1] = \frac{1}{1/2}[3-(1)(-1)] = 8$$

$$y_3 = \frac{1}{l_{33}}[c_3 - l_{32}y_2] = \frac{1}{4}[1-(-1)(8)] = \frac{9}{4}$$

- 변수 벡터 **y**의 값

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 8 \\ \frac{9}{4} \end{bmatrix}$$

- **Ux** = **y**의 형태

$$\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 8 \\ \frac{9}{4} \end{bmatrix}$$

다양한 상하행렬 분해

- 첫 번째 변수 벡터 **x**를 구하는 알고리즘

$$x_n = y_n \qquad \textbf{(A11)}$$

  - ✓ n은 행렬의 가장 큰 차수를 표시

- (A11)를 이용

$$x_3 = y_3 = \frac{9}{4}$$

- 나머지 변수들을 구하는 알고리즘

$$x_i = y_i - u_{i,i+1}x_{i+1} \qquad \textbf{(A12)}$$

  - ✓ 첨자 i는 2와 1을 순차적으로 이용

다양한 상하행렬 분해

➢ (A12)를 이용하여 나머지 변수들을 구하는 과정

$$x_2 = y_2 - u_{23}x_3 = 8 - (2)\left(\frac{9}{4}\right) = \frac{7}{2}$$

$$x_1 = y_1 - u_{12}x_2 = -1 - \left(-\frac{1}{2}\right)\left(\frac{7}{2}\right) = \frac{3}{4}$$

➢ 변수 벡터 $\mathbf{x}$의 값

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{7}{2} \\ \frac{9}{4} \end{bmatrix}$$

다양한 상하행렬 분해

참고 사항

➢ 제시한 알고리즘의 학습이 주목적이 아님

➢ 상하행렬 분해를 이용하는 과정의 알고리즘은 일반화시킨 형태가 아님

✓ 주어진 문제의 성질에 맞추어서 조금씩 수정이 필요

✓ 모든 상하행렬 문제에 사용은 불가

➢ 수치해석적으로 어떻게 알고리즘을 이용하는가에 대해 소개하는 정도

다양한 상하행렬 분해

예제 9 상하행렬 분해

상하행렬 분해 중 가우스 소거법과 같은 방법으로 푸는 첫 번째 방법을 이용하여, 다음에 주어지는 선형시스템의 변수 벡터 **x**를 구하라.

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 2 \\ 1 & 2 & -1 \\ -2 & -2 & 2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 7 \\ 2 \\ 0 \end{bmatrix}$$

Tip !

- ✓ $\mathbf{Ly} = \mathbf{c}$의 변수 벡터는 **y**를 전진 대입법을 이용하여 구함
- ✓ $\mathbf{Ux} = \mathbf{y}$의 변수 벡터는 **x**를 후진 대입법을 이용하여 구함

부분 피봇팅을 이용한 상하행렬 분해

- 부분 피봇팅을 사용하여 상하행렬 분해를 실행하는 식

  $\mathbf{LU} = \mathbf{PA}$ (20)

- **P**는 치환행렬(permutation matrix)을 표시
  - ✓ 어느 행 어느 열이나 단 한 개의 원소가 1
  - ✓ 다른 모든 원소가 0
  - ✓ 정방행렬
- 3 × 3 치환행렬의 예

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

부분 피봇팅을 이용한 상하행렬 분해

- 식 (20)의 우항 **PA**의 계산

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} A_{3,*} \\ A_{1,*} \\ A_{2,*} \end{bmatrix}$$

  - ✓ 행렬 **PA**는 행렬 **A**에서 행끼리 자리바꿈하여 구함
- 절댓값이 가장 큰 피봇 원소를 선택하기 위해서 행들의 자리바꿈이 필요
- 상수 벡터 **c**도 반드시 자리바꿈을 실행한 이후에 변수 벡터 **x**를 구함
- 선형시스템 **Ax** = **c**에 적용시킨 치환행렬 **P**

$$\mathbf{PAx} = \mathbf{Pc} \equiv \mathbf{d} \qquad \textbf{(21)}$$

부분 피봇팅을 이용한 상하행렬 분해

- 단위행렬이 치환행렬의 가장 기본적인 형태를 표시

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- 행렬 L과 U를 구하는 과정에서 자리바꿈을 두 번 했다고 가정
- 첫 번째 자리바꿈에서 1행과 3행을 바꾼 후에 다시 정렬된 치환행렬

$$\mathbf{P}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

- 두 번째 자리바꿈에서 2행과 3행을 바꾼 후에 다시 정렬된 치환행렬

$$\mathbf{P}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

부분 피봇팅을 이용한 상하행렬 분해

- 최종적으로 구한 치환행렬

$$\mathbf{P} = \mathbf{P}_2\mathbf{P}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

  - ✓ 자리바꿈을 두 번 하기 전, 각각의 치환행렬을 곱함
- 식 (15)와 식 (16)에 자리바꿈을 적용시킨 새로운 상수 벡터 $\mathbf{d}$를 이용

$$\begin{aligned} \mathbf{LUx} &= \mathbf{d} \\ \mathbf{Ly} &= \mathbf{d} \\ \mathbf{Ux} &= \mathbf{y} \end{aligned} \qquad (22)$$

부분 피봇팅을 이용한 상하행렬 분해

- **[예제 9]에서 주어진 선형시스템의 변수 벡터 $\mathbf{x}$의 실제 계산 예**
  - 1열에서 가장 큰 절댓값 원소는 −2
  - 부분 피봇팅을 이용하여 3행과 1행을 서로 자리 바꿈

$$\begin{bmatrix} -2 & -2 & 2 \\ 1 & 2 & -1 \\ -1 & 1 & 2 \end{bmatrix} \qquad (R1) \leftrightarrow (R3)$$

  - 1행 1열의 −2를 피봇 원소로 이용하여 구하는 2행과 3행의 승수들

$$m_{21} = \frac{a_{21}^{(1)}}{a_{11}^{(1)}} = -\frac{1}{2}$$

$$m_{31} = \frac{a_{31}^{(1)}}{a_{11}^{(1)}} = \frac{1}{2}$$

부분 피봇팅을 이용한 상하행렬 분해

- 1행에 $-\frac{1}{2}$을 곱하고 그 결과를 2행에서 뺀 뒤에 2행에 놓음
- 1행에 $\frac{1}{2}$을 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\begin{bmatrix} -2 & -2 & 2 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \quad \begin{matrix} \\ \leftarrow (R2) - \left(-\frac{1}{2}\right) \times (R1) \\ \leftarrow (R3) - \left(\frac{1}{2}\right) \times (R1) \end{matrix}$$

- 2열에서 가장 큰 절댓값 원소는 2
- 부분 피봇팅을 이용하여 2행과 3행을 자리바꿈

$$\begin{bmatrix} -2 & -2 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad (R2) \leftrightarrow (R3)$$

부분 피봇팅을 이용한 상하행렬 분해

- 2행 2열의 2를 피봇 원소로 이용하여 구하는 3행의 승수

$$m_{32} = \frac{a_{32}^{(2)}}{a_{22}^{(2)}} = \frac{1}{2}$$

- 2행에 $\frac{1}{2}$을 곱하고 그 결과를 3행에서 뺀 뒤에 3행에 놓음

$$\begin{bmatrix} -2 & -2 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix} \quad \leftarrow (R3) - \left(\frac{1}{2}\right) \times (R2)$$

- 첫 번째 자리바꿈에서 1행과 3행을 서로 바꿈
  - ✓ 자리바꿈할 승수들이 없기 때문에 더 이상 승수들의 자리바꿈은 불필요
- 두 번째 자리바꿈에서 2행과 3행을 서로 바꿈
  - ✓ 승수 m_{21}과 m_{31}의 자리바꿈이 필요

부분 피봇팅을 이용한 상하행렬 분해

➢ 식 (18)을 이용하여 구하는 행렬 **L**과 행렬 **U**

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} -2 & -2 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix}$$

➢ 최종적으로 사용할 치환행렬의 연산

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

✓ 첫 번째 자리바꿈에서 1행과 3행을 서로 바꿈

✓ 두 번째 자리바꿈에서 2행과 3행을 서로 바꿈

부분 피봇팅을 이용한 상하행렬 분해

➢ 자리바꿈을 적용시킨 새로운 상수 벡터 **d**

$$\mathbf{d} = \mathbf{Pc} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 7 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ 2 \end{bmatrix}$$

➢ 전진 대입법을 이용하여 구하는 **Ly** = **d**의 변수 벡터 **y**

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ 2 \end{bmatrix}$$

부분 피봇팅을 이용한 상하행렬 분해

➢ 변수 벡터 **y**의 값

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ -\frac{3}{2} \end{bmatrix}$$

➢ 후진 대입법을 이용하여 구하는 **Ux** = **y**의 변수 벡터 **x**

$$\begin{bmatrix} -2 & -2 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ -\frac{3}{2} \end{bmatrix}$$

➢ 상하행렬 분해를 이용하여 구한 변수 벡터 **x**의 최종 결과

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

부분 피봇팅을 이용한 상하행렬 분해

MATLAB 6 매트랩을 이용한 상하행렬 분해

[예제 9]의 결과를 매트랩을 이용하여 실행하라.

Tip !

✓ 상하행렬 분해를 할 수 있는 매트랩에서 제공하는 함수 이름은 **lu**

[L, U, P] = lu(A)

✓ **L**은 하삼각행렬, **U**는 상삼각행렬, **P**는 자리바꿈을 보여주는 치환행렬
✓ **A**는 주어진 계수행렬