연립 선형 대수 방정식

김기택
국민대학교 소프트웨어학과

연립 선형 대수 방정식

- n 개의 미지수를 가지고 있는 n 개의 선형 대수 방정식의 해를 구한다.
  - 이 방정식을 벡터식으로 표현하면 다음과 같다.

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

- 연립 선형 대수 방정식은 공학문제에서 대단히 많이 나타나고 있어 이의 효율적인 해결에 대한 중요성이 매우 높다.

- 종종 방정식의 규모가 수백 ~ 수천에 이를 때가 많아, 이를 효율적으로 계산하기 위한 많은 방법이 고안되었다.
  - 다양한 알고리즘이 만들어졌으며 그 중 잘 알려진 LAPACK (Linear Algebra PACKAGE)는 오래 전 Fortran77으로 작성되었다.

표기법

- 연립 대수 방정식은 다음과 같은 형식을 취한다.

$$A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n = b_1$$
$$A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n = b_2$$
$$\vdots$$
$$A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n = b_n$$

  - 여기서 계수 $\boldsymbol{A}_{ij}$와 상수 b_j는 알려져 있고 x_i는 미지수를 나타낸다.

- 이를 행렬 표기법으로 나타내면 다음과 같다.

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

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}$$

해의 유일성

- 연립 선형 대수 방정식이 고유한 해(unique solution)를 가지는 조건
  - 계수 행렬식(determinant)이 **가역행렬(nonsingular, 역행렬을 갖는 행렬)**이어야 함
  - 이는 $|A| \neq 0$ 인 경우이다.

- 가역행렬의 행과 열은 **선형 독립적(linearly independent)**이다.
  - 가역행렬의 어떤 행과 열도 다른 행(또는 열)의 선형 조합(linear combination)에 의해 생성되지 않음을 의미한다.

- 특이행렬(singular, 역행렬을 갖지 않는 행렬)인 경우 방정식은 상수 벡터에 따라 무한 수의 해 또는 전혀 해를 갖지 않을 수 있다.
  - 예) $2x + y = 3, 4x + 2y = 6$ 의 경우 두번째 식은 첫번째 식에 2를 곱하여 얻어진다. 이와 같은 경우 두 식은 서로 선형 독립적이지 않다. 해가 무한대로 많다.
  - 예) $2x + y = 3, 4x + 2y = 0$ 의 경우 두 식은 서로 모순된다. 이 경우 해는 없다.

불량 조건

- 계수 행렬이 거의 특이 행렬 일 때 (즉 $|\mathbf{A}|$ 가 매우 작을 때) 수치 계산에서 오차가 크게 발생하여 문제가 된다.
  - 계수 행렬이 얼마나 작은지를 판별할 수 있는 참조가 필요하다.
  - 이와 같은 참조를 **행렬의 놈(norm)** 이라 하며, $\|\mathbf{A}\|$ 로 표시한다.
  - 행렬의 놈이 다음과 같을 때 행렬식이 작다고 말한다.

$$|\mathbf{A}| << \|\mathbf{A}\|$$

놈의 종류

- 유클리드 놈(Euclidean norm)

$$\|\mathrm{A}\|_e = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^2}$$

- 행-합 놈(row-sum norm) 또는 무한 놈(infinity norm)

$$\|\mathrm{A}\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |A_{ij}|$$

- 행렬 조건수 (matrix condition number)

$$\mathrm{cond(A)} = \|\mathrm{A}\|\ \|\mathrm{A}^{-1}\|$$

  - 행렬 조건수가 1에 가까워지면 행렬이 잘 컨디셔닝된 것이다.
  - 특이 행렬일 때 행렬 조건수는 무한대가 된다.
  - 행렬 조건수를 계산할 때 많은 비용이 들기 때문에 행렬의 원소 크기와 행렬식을 비교하여 컨디셔닝을 측정하는 것으로 충분하다.

컨디셔닝이 잘못된 예

- 다음 방정식을 푼다고 하자.

$$2x + y = 3$$

$$2x + 1.001y = 0$$

  - 방정식의 해는 $x = 1501.5, y = -3000$ 이다.
- 계수 행렬의 행렬식은 $|A| = 2\,(1.001) - 2\,(1) = 0.002$ 이며 이는 계수보다 훨씬 작으므로, 방정식이 불완전하게 컨디셔닝 되었다. 이를 확인하는 방법으로 두번째 방정식을 다음과 같이 변경한 후 풀어 보면,

$$2x + 1.002y = 0$$

  - 결과는 $x = 751.5, y = -1500$ 이 되어 y 계수의 0.1% 의 변화가 해의 100% 변화를 만든다.
  - 불완전하게 컨디셔닝된 방정식의 수치해는 신뢰할 수 없다.
  - 수치 계산과정에서 반올림 오류를 피할 수 없게 된다.

- **계산을 하기 전 신뢰성을 위해 계수 행렬의 행렬식을 검토**해야 한다.

선형 시스템

- 선형 시스템은 다양한 공학 문제에서 발생하므로 매우 중요하게 다루어진다.
  - 구조해석, 탄성 고체, 열전달, 유체의 누출, 전자기장, 전기회로 등

- 불연속적 시스템
  - 트러스, 전기회로 등 정적인 시스템
  - 선형 대수방정식으로 직접 모델링이 된다.

- 연속적 시스템
  - 동역학, 유체역학 등 동적인 시스템은 미분방정식으로 구성된다.
  - 해를 구하기 위해 미분 방정식을 연립 대수 방정식으로 근사하여야 한다.
  - 이를 위해 유한 차분법, 유한 요소법 그리고 경계 요소법 등을 사용한다. 이를 **이산화 (discretization)** 이라 한다.
  - 이산화를 거친 상태는 종종 연립 선형대수 방정식이 되며 이를 푸는 문제가 된다.
  - 계수 행렬, $\mathbf{A}$ 는 시스템의 특성을, x 는 시스템의 응답, 그리고 b 는 외부 조건을 나타낸다.

해를 구하는 방법

- 직접법과 반복법의 두 가지 종류가 있다.

- **직접법**의 일반적인 특징은 기존 방정식을 등가 방정식(동일한 해를 가지는 방정식)으로 변환하는 것이다.
  - 변환을 위해서 다음 세 가지 조작을 적용한다.
  1. 두 방정식 교환 ($|\mathbf{A}|$ 부호 변경)
  2. 방정식에 0 이 아닌 상수 곱하기 (같은 상수로 $|A|$ 를 곱함)
  3. 방정식에 0 이 아닌 상수를 곱한 후 다른 방정식에서 빼기 ($|A|$ 는 변경되지 않음)

- **반복법**(간접법)에서는 처음 x 해를 추측으로 시작한 후 특정 수렴조건에 도달할 때까지 해를 반복적으로 수정한다.
  - 많은 수의 반복을 실행해야 하므로 직접법에 비해 효율적이지 않다.
  - 그러나, 행렬의 크기가 매우 크거나, 희소한 경우(sparse: 대부분의 계수가 0)는 계산 이점이 있다.

직접법 개요

- 가장 널리 사용되는 직접법은 다음과 같다.

방법	초기 형태	최종 행렬
가우스(Gauss) 소거	**A x = b**	**U x = e**
LU 분해	**A x = b**	**L U x = b**
Gauss-Jordan 소거	**A x = b**	**I x = c**

- U 는 상삼각 행렬(upper triangular matrix)

$$U = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

- L 은 하삼각 행렬(lower triangular matrix)

$$L = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix}$$

- I 는 항등 행렬(identity matrix)를 나타낸다.

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

삼각 행렬의 장점

- 많은 계산을 단순화 해준다.
- 예들 들어 $\mathbf{L}\,\mathbf{x} = \mathbf{c}$ 를 살펴 보자.

$$L_{11}\,x_1 = c_1$$
$$L_{21}\,x_1 + L_{22}\,x_2 = c_2$$
$$L_{31}\,x_1 + L_{32}\,x_2 + L_{33}\,x_3 = c_3$$

- 첫번째 방정식부터 풀면 각 방정식이 한번에 하나씩 미지수를 포함하기 때문에 해는 다음과 같이 진행될 것이다.

$$x_1 = c_1 / L_{11}$$
$$x_2 = (c_2 - L_{21}\,x_1) / L_{22}$$
$$x_3 = (c_3 - L_{31}\,x_1 - L_{32}\,x_2) / L_{33}$$

  - 이 절차를 전진 대입(forward substitution) 이라 한다.
  - 가우스 소거법에서는 유사한 방법으로 후진 대입(back substitution)을 통해 진행한다.
  - LU 분해법에서도 전진과 후진 대입을 모두 사용하고,
  - Gauss-Jordan 법에서는 이미 각 행이 고유한 해를 포함한다.

예제 2.1

다음 행렬이 특이 행렬인지 확인하시오.

$$\mathbf{A} = \begin{bmatrix} 2.1 & -0.6 & 1.1 \\ 3.2 & 4.7 & -0.8 \\ 3.1 & -6.5 & 4.1 \end{bmatrix}$$

[풀이] 행렬식을 구해 본다.

$$|\mathbf{A}| = 2.1 \begin{vmatrix} 4.7 & -0.8 \\ -6.5 & 4.1 \end{vmatrix} - (-0.6) \begin{vmatrix} 3.2 & -0.8 \\ 3.1 & 4.1 \end{vmatrix} + 1.1 \begin{vmatrix} 3.2 & 4.7 \\ 3.1 & -6.5 \end{vmatrix}$$

$$= 2.1\ (14.07) + 0.6\ (15.60) + 1.1\ (-35.37) = 0$$

행렬식이 0 이므로 특이 행렬이다. 행렬을 살펴 보면 (3행) = (3 × 1행) – (2행) 임을 알 수 있다. 이는 선형 독립적이지 못하다.

예제 2.2

다음 방정식 $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ 를 풀어 보아라.

$$\mathbf{A} = \begin{bmatrix} 8 & -6 & 2 \\ -4 & 11 & -7 \\ 4 & -7 & 6 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 28 \\ -40 \\ 33 \end{bmatrix}$$

단, 계수 행렬의 LU 분해는 다음과 같다.

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 2 & 0 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & -3 & 1 \\ 0 & 4 & -3 \\ 0 & 0 & 2 \end{bmatrix}$$

[풀이] 먼저 $\mathbf{L}\,\mathbf{y} = \mathbf{b}$ 방정식을 전진대입으로 해결한다.

$$2y_1 = 28 \qquad y_1 = 28/2 = 14$$

$$-y_1 + 2y_2 = -40 \qquad y_2 = (-40 + y_1) / 2 = (-40 + 14) / 2 = -13$$

$$y_1 - y_2 + y_3 = 33 \qquad y_3 = 33 - y_1 + y_2 = 33 - 14 - 13 = 6$$

예제 2.2 (continued)

그런 다음 해 x 는 후진 대입에 의해 $\mathbf{U}\,\mathbf{x} = \mathbf{y}$ 로부터 얻어진다.

$$2x_3 = y_3 \qquad x_3 = y_3/2 = 6/2 = 3$$

$$4x_2 - 3x_3 = y_2 \qquad x_2 = (y_2 + 3x_3) / 4 = [-13 + 3(3)] / 4 = -1$$

$$4x_1 - 3x_2 + x_3 = y_1 \qquad x_1 = (y_1 + 3x_2 - x_3) / 4 = [14 + 3(-1)] - 3] / 4 = 2$$

따라서 해는 $\mathbf{x} = [2 \;\; -1 \;\; 3]^T$ 가 된다.

이 예제는 LU 분해법이 실행되는 순서와 방법을 보여준다.

가우스 소거법

Gauss 소거법

- 가우스 소거법은 연립 방정식을 푸는 가장 친숙한 방법이다.
  - **소거 단계(elimination phase)**와 **후진 대입 단계(back substitution phase)** 로 구성된다.
  - 소거 단계의 기능은 방정식을 $\mathbf{U}\,\mathbf{x} = \mathbf{c}$ 의 형식으로 등가 변환하는 것이다. 이후 방정식은 후진 대입에 의해 해가 결정된다.
  - 다음 방정식을 풀어 보자.

$$4x_1 - 2x_2 + x_3 = 11 \qquad \text{(a)}$$

$$-2x_1 + 4x_2 - 2x_3 = -16 \qquad \text{(b)}$$

$$x_1 - 2x_2 + 4x_3 = 17 \qquad \text{(c)}$$

- 소거단계
  - 하나의 방정식에 상수를 곱하고 다른 방정식에서 빼는 방법을 적용한다.

$$\text{Eq. } (i) \leftarrow \text{Eq. } (i) - \lambda \times \text{Eq. } (j)$$

  - 빼는 방정식, Eq. (j) 를 **피벗 방정식(pivot equation)**이라 한다.

가우스 소거법 진행

- 위의 방정식들 중 Eq. (a) 를 피벗 방정식으로 하고 방정식 (b)와 (c) 로부터 x_1 을 소거하기 위해 승수 λ 를 선택한다.

$$\text{Eq. (b)} \leftarrow \text{Eq. (b)} - (-0.5) \times \text{Eq. (a)}$$

$$\text{Eq. (c)} \leftarrow \text{Eq. (c)} - 0.25 \times \text{Eq. (a)}$$

- 변환된 방정식은 다음과 같다.

$$4x_1 - 2x_2 + x_3 = 11 \qquad \text{(a)}$$

$$3x_2 - 1.5x_3 = -10.5 \qquad \text{(b)}$$

$$-1.5x_2 + 3.75x_3 = 14.25 \qquad \text{(c)}$$

  - 첫번째 단계가 완료되었다. 다음에는 피벗 방정식을 (b)로 하고 (c) 에서 x_2를 소거한다.

$$\text{Eq. (c)} \leftarrow \text{Eq. (c)} - (-0.5) \times \text{Eq. (b)}$$

  - 이 작업을 하면 첫번째와 두번째 방정식은 변화가 없고 세번째 방정식은 다음과 같다.

$$3x_3 = 9 \qquad \text{(c)}$$

  - 소거 단계가 완료 되었다. 등가방정식은 후진 대입으로 해를 구할 수 있다.

방정식의 행렬 표현

- 증대된(augmented) 계수행렬(coefficient matrix)는 계산 수행에 매우 편리한 도구이다. 원래의 방정식은 다음과 같이 표현된다.

$$\begin{bmatrix} 4 & -2 & 1 & 11 \\ -2 & 4 & -2 & -16 \\ 1 & -2 & 4 & 17 \end{bmatrix}$$

- 가우스 소거의 첫번째와 두번째 단계에 의해 생성된 등가 방정식은 다음과 같다.

$$\begin{bmatrix} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & -1.5 & 3.75 & 14.25 \end{bmatrix}$$

$$\begin{bmatrix} 4 & -2 & 1 & 11 \\ 0 & 3 & -1.5 & -10.5 \\ 0 & 0 & 3 & 9 \end{bmatrix}$$

- 방정식의 행렬식은 동일하다. 삼각 행렬의 행렬식은 매우 쉽게 구할 수 있다.

$$|\mathbf{A}| = |\mathbf{U}| = U_{11} \times U_{22} \times \ldots \times U_{nn}$$

후진 대입 단계

- 미지수는 후진 대입 단계로 구할 수 있다. 방정식 (c), (b), (a) 의 순으로 구한다.

$$x_3 = 9/3 = 3$$

$$x_2 = [-10.5 + 1.5(3)] / 3 = -2$$

$$x_1 = [11 + 2(-2) - 3] / 4 = 1$$

소거 단계 알고리즘 (1)

- 소거 단계에서 중간 지점의 방정식을 보면 계수 행렬의 첫번째 k 행이 이미 상삼각 형태로 변형되어 있다. 따라서 현재 피벗 방정식은 k 번째 방정식이며 그 아래의 모든 방정식은 변형된다. 소거 과정의 계수 행렬은 첫번째 행을 제외한 모든 행은 원래의 상태와 동일하지 않다. 이는 외부조건 벡터, $\mathbf{b}$ 도 동일하게 적용된다.

파이썬 인덱스 0

파이썬 인덱스 k-1

파이썬 인덱스 n-1

$$\left[\begin{array}{ccccccccc|c}
A_{11} & A_{12} & A_{13} & \cdots & A_{1k} & \cdots & A_{1j} & \cdots & A_{1n} & b_1 \\
0 & A_{22} & A_{23} & \cdots & A_{2k} & \cdots & A_{2j} & \cdots & A_{2n} & b_2 \\
0 & 0 & A_{33} & \cdots & A_{3k} & \cdots & A_{3j} & \cdots & A_{3n} & b_3 \\
\vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\
0 & 0 & 0 & \cdots & A_{kk} & \cdots & A_{kj} & \cdots & A_{kn} & b_k \\
\hline
\vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\
0 & 0 & 0 & \cdots & A_{ik} & \cdots & A_{ij} & \cdots & A_{in} & b_i \\
\vdots & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots \\
0 & 0 & 0 & \cdots & A_{nk} & \cdots & A_{nj} & \cdots & A_{nn} & b_n
\end{array}\right]$$

← 피벗행

← 변형된 피벗행

소거 단계 알고리즘 (2)

- i 번째 행을 변형될 피벗 방정식 아래의 행으로 놓는다. 즉, A_{ik} 요소가 소거된다.
- 피벗행에 $\lambda = A_{ik} / A_{kk}$ 를 곱하고 이를 i 번째 행에서 뺀다. i 번째 행의 변경된 상태는 다음과 같다.

$$A_{ij} \leftarrow A_{ij} - \lambda A_{kj}\ ,\ \ j = k, k+1, \ldots , n$$
$$b_i \leftarrow b_i - \lambda b_k$$

- 전체 행렬을 상삼각 형태로 변환하기 위해 위의 방정식에서 k 와 i 의 범위를 구한다.
  - $k = 1, 2, \ldots, n\text{-}1$ (피벗행 선택, 총 $n-1$ 개 행, 마지막 행은 피벗행이 되지 않음)
  - $i = k+1, k+2, \ldots, n$ (변환될 행 선택, 총 $n - k$ 개 행) 이 된다.
  - 이를 프로그램으로 나타내면 다음과 같다.

```
for k in range(0, n−1):
    for i in range(k+1, n):
        if a[i, k] != 0.0:
            lam = a[i, k] / a[k, k]
            a[i, k+1:n] = a[i, k+1:n] – lam * a[k, k+1:n]
            b[i] = b[i] – lam * b[k]
```

파이썬은 0부터 인수가 시작한다. (총 $n-1$ 개)

$k+1$ 부터 시작하므로 A_{ik} 가 0으로 대체 되지 않고 그대로 남는다. ($k+1, \ldots, n\text{-}1$)

A_{ik} 이 0 이면 i 행의 변환을 건너 뛴다.

벡터화 연산 (for 루프 없는 방법)

후진 대입 단계 알고리즘

- 가우스 소거단계 이후 증대된 계수 행렬은 다음과 같아진다.

$$[\mathbf{A} \quad |\mathbf{b}] = \left[\begin{array}{ccccc|c} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} & b_1 \\ 0 & A_{22} & A_{23} & \cdots & A_{2n} & b_2 \\ 0 & 0 & A_{33} & \cdots & A_{3n} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nn} & b_n \end{array}\right]$$

- 마지막 방정식 $A_{nn}\, x_n = b_n$ 을 먼저 풀면, $x_n = b_n / A_{nn}$ 이 된다.
- 다음과 같은 k 번째 방정식에서 x_k 를 구하면 다음과 같다.

$$A_{kk}\, x_k + A_{k,k+1}\, x_{k+1} + \cdots + A_{kn}\, x_n = b_k$$

$$x_k = \left(b_k - \sum_{j=k+1}^{n} A_{kj} x_j \right) \frac{1}{A_{kk}}, \qquad k = n-1, n-2, \ldots, 1$$

  - 프로그램으로 나타내면 다음과 같다.

```
for k in range(n−1, −1, −1):
    x[k] = (b[k] − dot(a[k, k+1: n], x[k+1: n])) / a[k,k]
```

n-1 부터 1씩 감소하면서 0 까지 (-1 보다 하나 큰 수)

벡터화 연산 (for 루프 없는 방법)

가우스 소거법의 연산 횟수

- 알고리즘의 실행 시간은 긴 연산의 곱셈과 나눗셈의 수에 크게 의존한다.

- 가우스 소거법에서는 **소거 단계에서 약 $n^3/3$** 과 같은 연산이 포함되며,
  후진 대입 단계에서는 $n^2/2$ 연산이 포함된다.
  - 대부분 **소거 단계에서 시간이 소요**됨을 알 수 있다. 이는 방정식의 수에 따라 빠르게 증가한다.

가우스 소거법 프로그램

```
## module gaussElimin
import numpy as np
''' x = haussElim(a,b)
    solves [a] b = x by Gauss Elimination
'''
def gaussElimin(a,b):    # textbook code
    n = len(b)               # length of vector
    # Elimination phase
    for k in range(0, n-1):
        for i in range(k+1, n):
            if a[i, k] != 0.0:
                lam = a[i, k] / a[k, k]       # Calculate constant for elimination
                a[i, k+1:n] = a[i, k+1:n] - lam * a[k, k+1:n]
                b[i] = b[i] - lam*b[k]
    # Back substitution
    for k in range(n-1,-1,-1):
        b[k] = (b[k]-np.dot(a[k,k+1:n],b[k+1:n]))/a[k,k]
    return b
```

모듈에 대한 설명

함수 정의

해가 b 에 저장됨

결과 b 를 반환

방정식의 다중 세트

- 계수 행렬은 변함이 없으나 외부조건 벡터, **b** 가 하나가 아닌 복수벡터일 때 해도 복수의 벡터로 나타난다.

$$\mathbf{A}\,\mathbf{X} = \mathbf{B}$$

$$\mathbf{X} = [\mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \ldots \mathbf{x}_m]\,(n \times m)\,, \quad \mathbf{B} = [\mathbf{b}_1 \;\; \mathbf{b}_2 \;\; \ldots \mathbf{b}_m]\,(n \times m)$$

- 이와 같은 다중 세트를 가우스 소거법으로 처리할 때, 증강 계수 행렬에 모든 외부조건 벡터를 포함하여 계수 행렬과 동시에 소거 처리하고 후진 대입을 통해 해를 구한다.
  - 다음 예제에서 살펴 본다.

예제 2.3

다음 식 **A X** = **B** 을 가우스 소거법을 사용하여 해를 구하라.

$$\mathbf{X} = \begin{bmatrix} 6 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 6 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -14 & 22 \\ 36 & -18 \\ 6 & 7 \end{bmatrix}$$

[풀이] 증대된 계수 행렬은 다음과 같다.

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ -4 & 6 & -4 & 36 & -18 \\ 1 & -4 & 6 & 6 & 7 \end{array}\right]$$

소거 단계는 다음 두 단계로 구성된다.

$$\text{row } 2 \leftarrow \text{row } 2 + (2/3) \times \text{row } 1$$

$$\text{row } 3 \leftarrow \text{row } 3 - (1/6) \times \text{row } 1$$

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & -10/3 & 35/6 & 25/3 & 10/3 \end{array}\right]$$

예제 2.3 (continued)

그리고

$$\text{row } 3 \leftarrow \text{row } 3 + \text{row } 2$$

$$\left[\begin{array}{ccc|cc} 6 & -4 & 1 & -14 & 22 \\ 0 & 10/3 & -10/3 & 80/3 & -10/3 \\ 0 & 0 & 5/2 & 35 & 0 \end{array}\right]$$

후진 대입 단계에서 우선 $\mathbf{x}_1$ 을 계산한다.

$$X_{31} = \frac{35}{\frac{5}{2}} = 14, X_{21} = \frac{\frac{80}{3} + \left(\frac{10}{3}\right) X_{31}}{\frac{10}{3}} = 22, X_{11} = \frac{-14 + 4X_{21} - X_{31}}{6} = 10$$

두번째 해 벡터, $\mathbf{x}_2$ 는 다음과 같이 계산된다.

$$X_{32} = 0, X_{22} = \frac{-\frac{10}{3} + \left(\frac{10}{3}\right) X_{32}}{\frac{10}{3}} = -1, X_{12} = \frac{22 + 4X_{22} - X_{32}}{6} = 3$$