수치해석 과제

과제번호: 12

과제제목: Gaussian Elimination

제 출 날 짜: 2022년 11월 23일
학　　과: 소프트웨어학부
학　　번: 2020203081
이　　름: 심유미

1. 과제 설명

Equation은 변수끼리 관계있는 식을 의미하고, linear system은 이 equation들이 선형으로 이루어져 있을 때를 말한다. 가우스 소거법은 어떤 선형 시스템의 솔루션을 구할 때 사용된다. Equation E_i, E_j가 있을 때, 다음 3가지 연산으로 가우스 소거법을 진행할 수 있다.

① $(\lambda E_i) \rightarrow (E_i)$
: E_i는 0이 아닌 상수 λ와 곱해질 수 있고, 그 결괏값은 E_i를 대체할 수 있다.
② $(E_i + \lambda E_j) \rightarrow (E_i)$
: 상수 λ에 대해 λE_j는 E_i와 더해질 수 있고, 그 결괏값은 E_i를 대체할 수 있다.
③ $(E_i) \leftrightarrow (E_j)$
: E_i와 E_j는 순서를 서로 바꿀 수 있다.

위 3개의 연산을 하더라도 equation의 해는 영향을 받지 않는다. n개의 equation에 대해 가우스 소거법을 마치면 E_n부터 E_1까지 순서대로, 해 x_n부터 x_1을 순서대로 구할 수 있다. 이 과정을 backward substitution이라고 한다.

Linear system에는 대응되는 augmented matrix가 존재한다. 여기서 augmented matrix는 다음을 의미한다.

$$
\begin{aligned}
E_1:&\quad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1,\\
E_2:&\quad a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2,\\
&\qquad\qquad \vdots\\
E_n:&\quad a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n,
\end{aligned}
$$

그림 1) Equation E_1, ..., E_n

$$
\tilde{A} = [A, \mathbf{b}] = \left[\begin{array}{cccc:c}
a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1}\\
a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1}\\
\vdots & \vdots & & \vdots & \vdots\\
a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1}
\end{array}\right]
$$

그림 2) 그림 1에 대응하는 Augmented matrix

Augmented matrix만으로도 linear system에 대한 가우스 소거법을 진행할 수 있다. 우선 어느 i번째 row에서 현재 첫 번째로 있는 요소를 pivot point라 가정한다. i번째 row의 밑에 있는 row들에 대해, pivot point와 동일한 column에 있는 값들을 0으로 소거시킬 것이다. $(E_k - m_{ki}E_i) \rightarrow (E_k)$ (k = i + 1, ..., n) 연산을 적용할 것이고, 이때 m_{ki}는 a_{ki}/a_{ii}를 의미한다. 연산을 반복 적용하고 난 결과 augmented matrix를 통해 triangular linear system을 얻을 수 있고, 이에 대해 backward substitution을 적용하면 해를 얻을 수 있다.

Pivot point가 0인 경우도 존재하는데, pivot point는 0이 되면 안 되므로 이 경우 동일한 column에 있는 값이 0이 아닌 다른 row와 현재 row의 위치를 바꿔줘야 한다.

가우스 소거법의 오차를 줄이기 위해 소거 이전에 다른 연산을 진행할 수 있다. 그중 하나인 partial pivoting은 동일한 column 내에서 pivot point의 절댓값이 가장 크길 원하는 방식이다. 만약 현재의 pivot point보다 다른 row에 있는 값이 더 크다면, 현재 row와 위치를 바꿔준다.

Scaled partial pivoting은 scale factor s_k를 이용하는 방식이다. s_k는 k번째 row에서 절댓값이 가장 큰 수를 의미한다. row의 모든 값을 각각의 s_k로 나눴다고 가정하고, 동일한 i번째 column에서 가장 큰 값을 보유하고 있는 p번째 row를 선택한다. a_{pi}를 pivot point로 사용할 것이므로, 현재의 row와 p번째 pow의 위치를 바꿔준다. 식으로 정리하면 다음과 같다.

$$s_k = \max_{i \le j \le n} |a_{kj}|$$

$$|a_{pi}| / s_p = \max_{i \le k \le n} |a_{ki}| / s_k$$

2. 소스 코드

```
#include <stdio.h>
#include <math.h>

void printResult(double aug[3][4]) {
	printf("a1\ta2\ta3\tb\n");
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 4; j++)
			printf("%.1f\t", aug[i][j]);
		printf("\n");
	}
}

void changeRow(double (&row1)[4], double (&row2)[4]) {
	double temp[4] = { 0, };

	for (int i = 0; i < 4;i++) {
		temp[i] = row1[i];
		row1[i] = row2[i];
		row2[i] = temp[i];
	}
}

void Q1_standardGaussian(double aug[3][4]) {
	// Gaussian elimination about ar1
	for (int r = 0; r < 3; r++) {

		// if pivot point == 0
		if (aug[r][r] == 0) {
```

```
                        // (Er) ↔ (Er+1)
                        // Re-operation r-th row
                        changeRow(aug[r], aug[r + 1]);
                        r--;
                        continue;
                }

                // Calculate mkr
                for (int k = r + 1; k < 3; k++) {
                        double mkr = aug[k][r] / aug[r][r];

                        // (Ek - mkrEr) → (Ek)
                        for (int c = r; c < 4; c++)
                                aug[k][c] -= mkr * aug[r][c];
                }
        }
        printResult(aug);
}

void Q2_partialPivoting(double aug[3][4]) {
        for (int r = 0; r < 3; r++) {

                // Check what is largest in same column
                int row = r;
                for (int R = r+1; R < 3; R++)
                        if (fabs(aug[row][r]) < fabs(aug[R][r])) row = R;

                // if aii is larger that aji
                if (row != r) {
                        // (Er) ↔ (ER)
                        // Re-operation r-th row
                        changeRow(aug[r], aug[row]);
                        r--;
                        continue;
                }

                // Elimination
                if (aug[r][r] == 0) {
                        changeRow(aug[r], aug[r + 1]);
                        r--;
                        continue;
```

```
                }
                for (int k = r + 1; k < 3; k++) {
                        double mkr = aug[k][r] / aug[r][r];
                        for (int c = r; c < 4; c++)
                                aug[k][c] -= mkr * aug[r][c];
                }
        }
        printResult(aug);
}

void Q3_scaledPartial(double aug[3][4]) {
        for (int r = 0; r < 3; r++) {
                // Set scale factor
                double sk[3] = { fabs(aug[r][r]), fabs(aug[r][r]), fabs(aug[r][r]) };
                for (int i = r; i < 3; i++)
                        for (int j = r; j < 3; j++)
                                if (sk[i] < fabs(aug[i][j])) sk[i] = fabs(aug[i][j]);

                // Find p-th row
                int p = r;
                for (int P = p + 1;P < 3;P++)
                        if (fabs(aug[p][r]) / sk[p] < fabs(aug[P][r] / sk[P])) p = P;

                // (Er) ↔ (EP)
                if (r != p)        changeRow(aug[r], aug[p]);

                // Elimination
                if (aug[r][r] == 0) {
                        changeRow(aug[r], aug[r + 1]);
                        r--;
                        continue;
                }
                for (int k = r + 1; k < 3; k++) {
                        double mkr = aug[k][r] / aug[r][r];
                        for (int c = r; c < 4; c++)
                                aug[k][c] -= mkr * aug[r][c];
                }
        }
        printResult(aug);
}
```

```
int main() {
	double a1_aug[3][4] = { {1, -5, 1, 7}, {10, 0, 20, 6}, {5, 0, -1, 4} };
	double b1_aug[3][4] = { {1, 1, -1, 1}, {1, 1, 4, 2}, {2, -1, 2, 3} };
	double c1_aug[3][4] = { {2, -3, 2, 5}, {-4, 2, -6, 14}, {2, 2, 4, 8} };
	double d1_aug[3][4] = { {0, 1, 1, 6}, {1, -2, -1, 4}, {1, -1, 1, 5} };

	printf("Q1. standard Gaussian elimination");
	printf("\na.\n"); Q1_standardGaussian(a1_aug);
	printf("\nb.\n"); Q1_standardGaussian(b1_aug);
	printf("\nc.\n"); Q1_standardGaussian(c1_aug);
	printf("\nd.\n"); Q1_standardGaussian(d1_aug);

	double a2_aug[3][4] = { {1, -5, 1, 7}, {10, 0, 20, 6}, {5, 0, -1, 4} };
	double b2_aug[3][4] = { {1, 1, -1, 1}, {1, 1, 4, 2}, {2, -1, 2, 3} };
	double c2_aug[3][4] = { {2, -3, 2, 5}, {-4, 2, -6, 14}, {2, 2, 4, 8} };
	double d2_aug[3][4] = { {0, 1, 1, 6}, {1, -2, -1, 4}, {1, -1, 1, 5} };

	printf("\nQ2. Gaussian elimination with partial pivoting");
	printf("\na.\n"); Q2_partialPivoting(a2_aug);
	printf("\nb.\n"); Q2_partialPivoting(b2_aug);
	printf("\nc.\n"); Q2_partialPivoting(c2_aug);
	printf("\nd.\n"); Q2_partialPivoting(d2_aug);

	double a3_aug[3][4] = { {1, -5, 1, 7}, {10, 0, 20, 6}, {5, 0, -1, 4} };
	double b3_aug[3][4] = { {1, 1, -1, 1}, {1, 1, 4, 2}, {2, -1, 2, 3} };
	double c3_aug[3][4] = { {2, -3, 2, 5}, {-4, 2, -6, 14}, {2, 2, 4, 8} };
	double d3_aug[3][4] = { {0, 1, 1, 6}, {1, -2, -1, 4}, {1, -1, 1, 5} };

	printf("\nQ3. Gaussian elimination with scaled partial pivoting");
	printf("\na.\n"); Q3_scaledPartial(a3_aug);
	printf("\nb.\n"); Q3_scaledPartial(b3_aug);
	printf("\nc.\n"); Q3_scaledPartial(c3_aug);
	printf("\nd.\n"); Q3_scaledPartial(d3_aug);
}
```

3. 실행 결과

```
Q1. standard Gaussian elimination
a.
a1      a2      a3      b
1.0     -5.0    1.0     7.0
0.0     50.0    10.0    -64.0
0.0     0.0     -11.0   1.0

b.
a1      a2      a3      b
1.0     1.0     -1.0    1.0
0.0     -3.0    4.0     1.0
0.0     0.0     5.0     1.0

c.
a1      a2      a3      b
2.0     -3.0    2.0     5.0
0.0     -4.0    -2.0    24.0
0.0     0.0     -0.5    33.0

d.
a1      a2      a3      b
1.0     -2.0    -1.0    4.0
0.0     1.0     1.0     6.0
0.0     0.0     1.0     -5.0
```

```
Q2. Gaussian elimination with partial pivoting
a.
a1      a2      a3      b
10.0    0.0     20.0    6.0
0.0     -5.0    -1.0    6.4
0.0     0.0     -11.0   1.0

b.
a1      a2      a3      b
2.0     -1.0    2.0     3.0
0.0     1.5     3.0     0.5
0.0     0.0     -5.0    -1.0

c.
a1      a2      a3      b
-4.0    2.0     -6.0    14.0
0.0     3.0     1.0     15.0
0.0     0.0     -0.3    22.0

d.
a1      a2      a3      b
1.0     -2.0    -1.0    4.0
0.0     1.0     1.0     6.0
0.0     0.0     1.0     -5.0
```

```
Q3. Gaussian elimination with scaled partial pivoting
a.
a1      a2      a3      b
5.0     0.0     -1.0    4.0
0.0     -5.0    1.2     6.2
0.0     0.0     22.0    -2.0

b.
a1      a2      a3      b
1.0     1.0     -1.0    1.0
0.0     -3.0    4.0     1.0
0.0     0.0     5.0     1.0

c.
a1      a2      a3      b
2.0     -3.0    2.0     5.0
0.0     -4.0    -2.0    24.0
0.0     0.0     -0.5    33.0

d.
a1      a2      a3      b
1.0     -1.0    1.0     5.0
0.0     1.0     1.0     6.0
0.0     0.0     -1.0    5.0
```