Xilinx Zynq FPGA, TI DSP, MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)
gcccompil3r@gmail.com
학생 - 이유성
dbtjd1102@naver.com

목차

*정방 행렬 , 대각 행렬, 단위 행렬 ,전치 행렬

*행렬 연산 (덧셈 , 뺄셈 , 곱)

*역행렬 -> 가우스 조르단 소거법(컴퓨터 성능상 좋음) ,Determinant (정석)

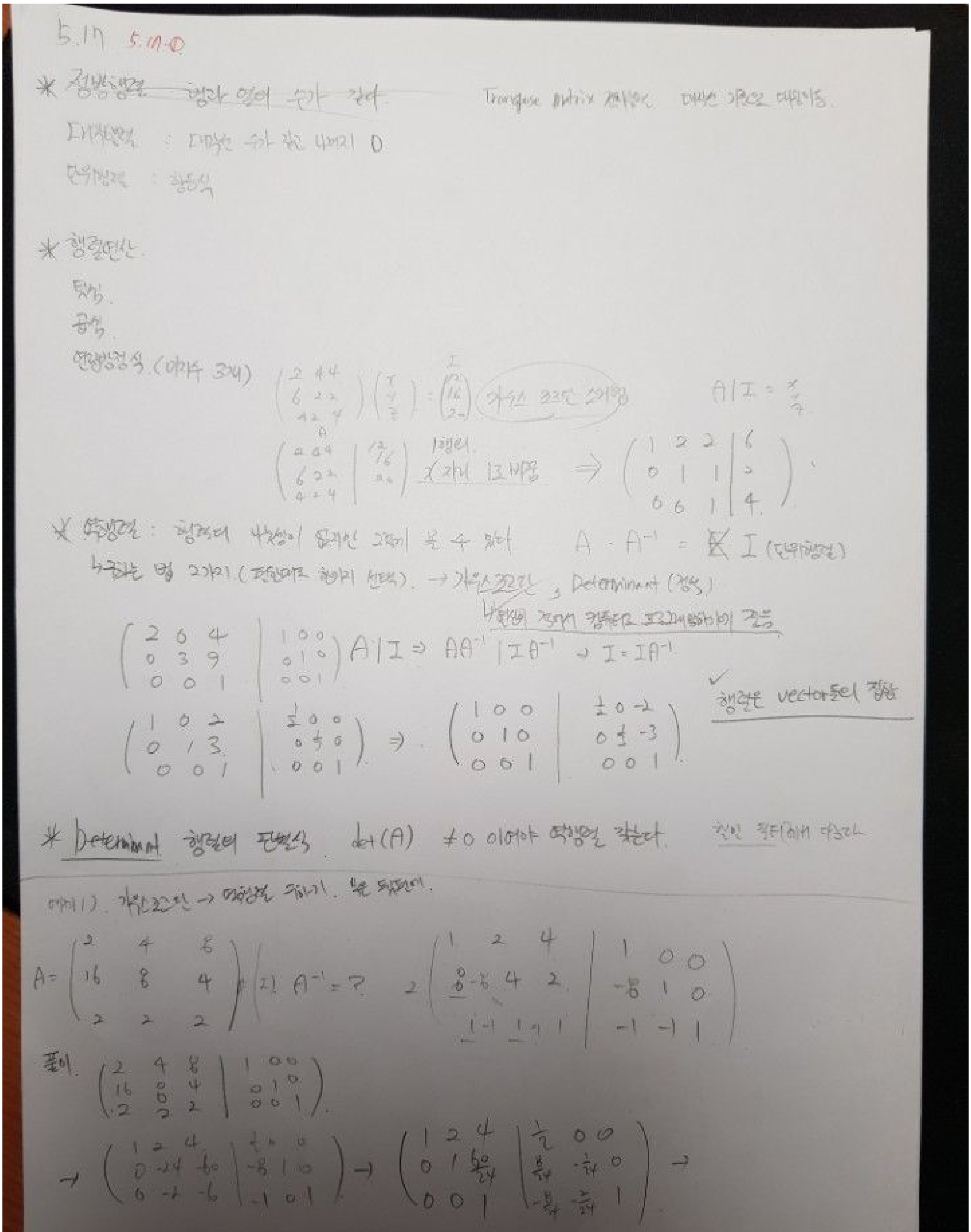
*Determinant = 행렬의 판별식 det(A) = 0 ->역행렬 없다

*행렬은 vector들의 집합

*크래머 공식(사람이 하기에 좋음)

==> 프로그래밍 하자

for문 안쓰는 이유 -> 성능이 더 좋다



5.17 5.17-①

* ~~정방행렬 : 행과 열의 수가 같다.~~ Transpose matrix 전치행렬 대각선 기준으로 대칭이동.

대각행렬 : 대각선 수가 값 나머지 0

단위행렬 : 항등식

* 행렬연산.

덧셈.

곱셈.

연립방정식.(미지수 3개)

$$\begin{pmatrix} 2 & 4 & 4 \\ 6 & 2 & 2 \\ 4 & 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 12 \\ 16 \\ 20 \end{pmatrix}$$

가우스 조르단 소거법 $A|I = \begin{matrix} x \\ y \\ z \end{matrix}$

$$\left(\begin{array}{ccc|c} 2 & 4 & 4 & 12 \\ 6 & 2 & 2 & 16 \\ 4 & 2 & 4 & 20 \end{array}\right)$$

1행의 ½ 곱해 1로 바꿈

$$\Rightarrow \left(\begin{array}{ccc|c} 1 & 2 & 2 & 6 \\ 0 & 1 & 1 & 2 \\ 0 & 6 & 1 & 4 \end{array}\right)$$

* 역행렬 : 행렬의 나눗셈이 필요한 경우 볼 수 있다 $A \cdot A^{-1} = \cancel{E}\ I$ (단위행렬)

구하는 법 2가지 (편한대로 하나 선택). → 가우스조르단, Determinant (공식)

나중에 가서 컴퓨터로 프로그래밍하여 구함

$$\left(\begin{array}{ccc|ccc} 2 & 0 & 4 & 1 & 0 & 0 \\ 0 & 3 & 9 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right) A|I \Rightarrow AA^{-1}|IA^{-1} \rightarrow I = IA^{-1}$$

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 2 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 3 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{1}{2} & 0 & -2 \\ 0 & 1 & 0 & 0 & \frac{1}{3} & -3 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

✓ 행렬은 vector들의 집합

* Determinant 행렬의 판별식. $\det(A) \neq 0$ 이어야 역행렬 갖는다. 칠판 필기에서 다르다

예제1). 가우스조르단 → 역행렬 구하기. 풀고 확인해.

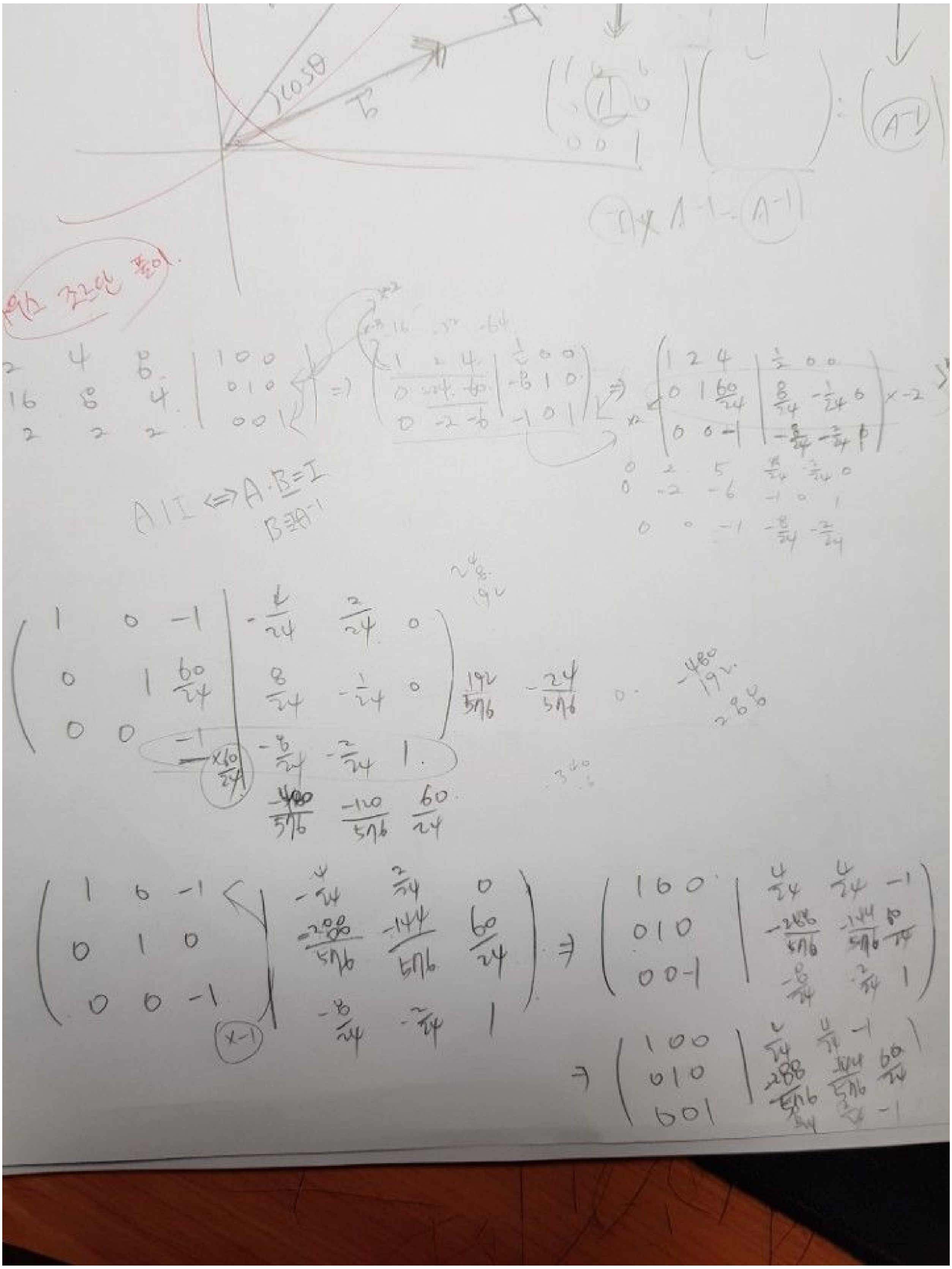
$$A = \begin{pmatrix} 2 & 4 & 8 \\ 16 & 8 & 4 \\ 2 & 2 & 2 \end{pmatrix} \text{의 } A^{-1} = ?$$

$$2\left(\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 8-8 & 4 & 2 & -8 & 1 & 0 \\ 1-1 & 1-1 & 1 & -1 & -1 & 1 \end{array}\right)$$

풀이.

$$\left(\begin{array}{ccc|ccc} 2 & 4 & 8 & 1 & 0 & 0 \\ 16 & 8 & 4 & 0 & 1 & 0 \\ 2 & 2 & 2 & 0 & 0 & 1 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{ccc|ccc} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & -24 & -60 & -8 & 1 & 0 \\ 0 & -2 & -6 & -1 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|ccc} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{60}{24} & \frac{8}{24} & -\frac{1}{24} & 0 \\ 0 & 0 & 1 & -\frac{8}{24} & \frac{2}{24} & 1 \end{array}\right) \rightarrow$$



$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \boxed{I} & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \ \end{pmatrix} = \begin{pmatrix} \boxed{A^{-1}} \end{pmatrix}$$

$\boxed{I} \times A^{-1} = \boxed{A^{-1}}$

우스 조르단 풀이.

$$\left(\begin{array}{ccc|ccc} 2 & 4 & 8 & 1 & 0 & 0 \\ 16 & 8 & 4 & 0 & 1 & 0 \\ 2 & 2 & 2 & 0 & 0 & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|ccc} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & -24 & -60 & -8 & 1 & 0 \\ 0 & -2 & -6 & -1 & 0 & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|ccc} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{60}{24} & \frac{8}{24} & -\frac{1}{24} & 0 \\ 0 & 0 & -1 & -\frac{8}{24} & -\frac{2}{24} & 1 \end{array}\right) \times -2$$

×2

×-8 -16 -32 -64

×2

$$\begin{array}{cccccc} 0 & 2 & 5 & \frac{8}{24} & -\frac{2}{24} & 0 \\ 0 & -2 & -6 & -1 & 0 & 1 \\ 0 & 0 & -1 & -\frac{8}{24} & -\frac{2}{24} & \end{array}$$

$A|I \Leftrightarrow A \cdot \underline{B} = I$

$B = A^{-1}$

248

192

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -1 & -\frac{4}{24} & \frac{2}{24} & 0 \\ 0 & 1 & \frac{60}{24} & \frac{8}{24} & -\frac{1}{24} & 0 \\ 0 & 0 & -1 & -\frac{8}{24} & -\frac{2}{24} & 1 \end{array}\right) \quad \frac{192}{576} \quad -\frac{24}{576} \quad 0$$

$\times\frac{60}{24}$

$$-\frac{480}{576} \quad -\frac{120}{576} \quad \frac{60}{24}$$

-480

192

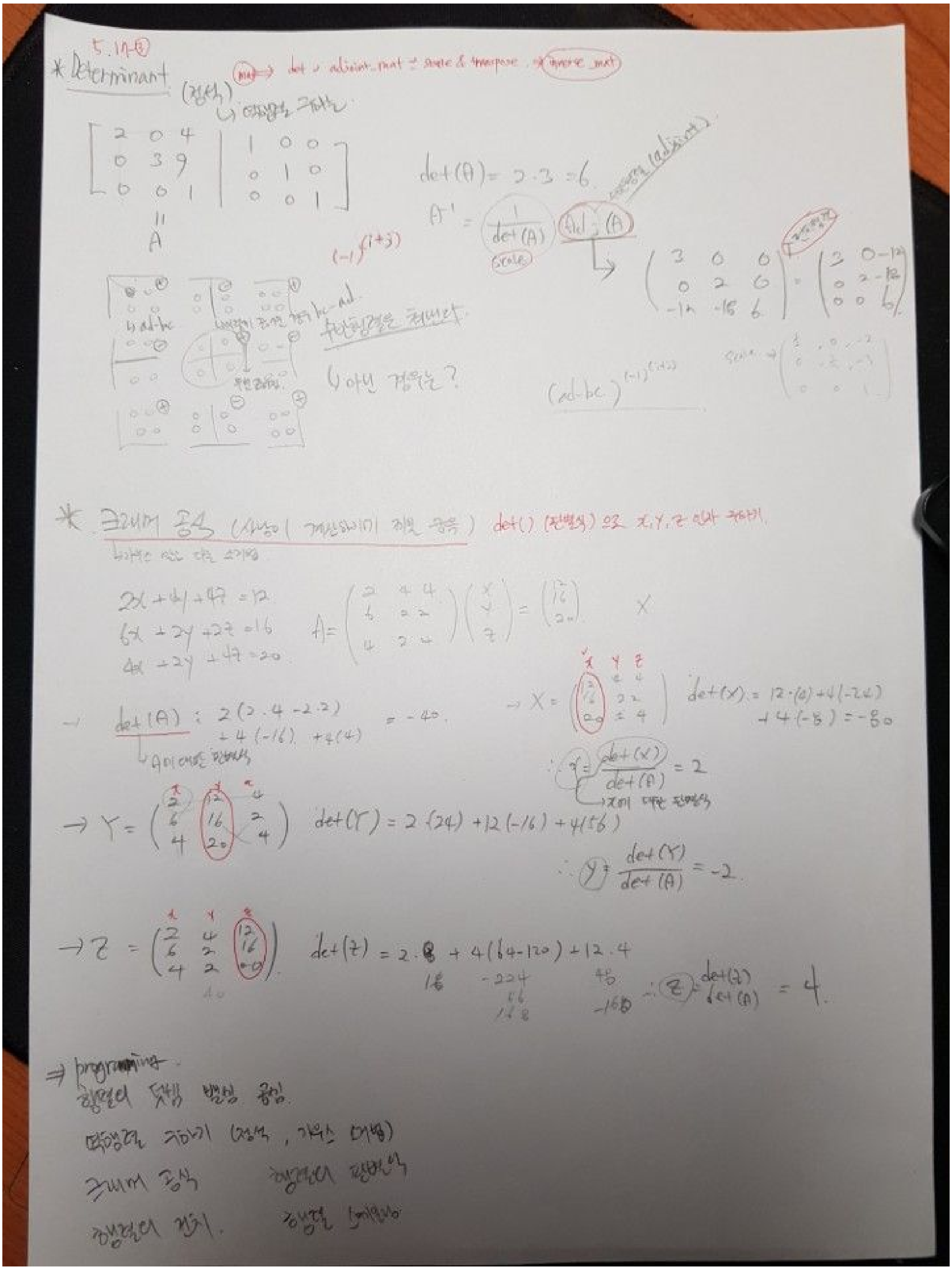
288

360

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -1 & -\frac{4}{24} & \frac{2}{24} & 0 \\ 0 & 1 & 0 & -\frac{288}{576} & -\frac{144}{576} & \frac{60}{24} \\ 0 & 0 & -1 & -\frac{8}{24} & -\frac{2}{24} & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{4}{24} & \frac{4}{24} & -1 \\ 0 & 1 & 0 & -\frac{288}{576} & -\frac{144}{576} & \frac{60}{24} \\ 0 & 0 & -1 & -\frac{8}{24} & -\frac{2}{24} & 1 \end{array}\right)$$

$\boxed{\times -1}$

$$\Rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{4}{24} & \frac{4}{24} & -1 \\ 0 & 1 & 0 & -\frac{288}{576} & -\frac{144}{576} & \frac{60}{24} \\ 0 & 0 & 1 & \frac{8}{24} & \frac{2}{24} & -1 \end{array}\right)$$



5.17-②

* Determinant (정석) → det → adjoint_mat → scale & transpose → inverse_mat

↳ 역행렬을 구하는

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 4 & 1 & 0 & 0 \\ 0 & 3 & 9 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

$\overset{\shortparallel}{A}$

$\det(A) = 2 \cdot 3 = 6$

$A^{-1} = \dfrac{1}{\det(A)} \operatorname{Adj}(A)$

scale

여인수행렬 (adjoint)

$(-1)^{(1+3)}$

$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ -12 & -18 & 6 \end{pmatrix}^{T} = \begin{pmatrix} 3 & 0 & -12 \\ 0 & 2 & -18 \\ 0 & 0 & 6 \end{pmatrix}$$

전치행렬

↳ ad-bc

내부에서 조합된 경우 bc-ad

[illegible] 부호변환

부호 교환

↳ 아닌 경우는?

$(ad-bc)^{(-1)^{(i+j)}}$

scale →

* 크래머 공식 (사람이 계산하기 제일 좋음) det() (판별식) 으로 x, y, z 값 구하기

↳ 가우스 없는 대신 소거법

$2x + 4y + 4z = 12$
$6x + 2y + 2z = 16$
$4x + 2y + 4z = 20$

$$A = \begin{pmatrix} 2 & 4 & 4 \\ 6 & 2 & 2 \\ 4 & 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 12 \\ 16 \\ 20 \end{pmatrix}$$

X

→ $\det(A) = 2(2 \cdot 4 - 2 \cdot 2) + 4(-16) + 4(4) = -40$

↳ A에 대한 판별식

$$\rightarrow X = \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 12 & 4 & 4 \\ 16 & 2 & 2 \\ 20 & 2 & 4 \end{pmatrix} \end{array}$$

$\det(X) = 12 \cdot (4) + 4(-24) + 4(-8) = -80$

$\therefore x = \dfrac{\det(X)}{\det(A)} = 2$

↳ X에 대한 판별식

$$\rightarrow Y = \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 2 & 12 & 4 \\ 6 & 16 & 2 \\ 4 & 20 & 4 \end{pmatrix} \end{array}$$

$\det(Y) = 2(24) + 12(-16) + 4(56)$

$\therefore y = \dfrac{\det(Y)}{\det(A)} = -2$

$$\rightarrow Z = \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 2 & 4 & 12 \\ 6 & 2 & 16 \\ 4 & 2 & 20 \end{pmatrix} \end{array}$$

40

$\det(Z) = 2 \cdot 8 + 4(64 - 120) + 12 \cdot 4$

16   −224   48
      56   −160
      128

$\therefore z = \dfrac{\det(Z)}{\det(A)} = 4$

⇒ programming

행렬의 덧셈 뺄셈 곱셈

역행렬 구하기 (정석, 가우스 소거법)

크래머 공식   행렬의 판별식

행렬의 전치.   행렬 Scaling

목차 프로그래밍 (가우스 조르단 제외)

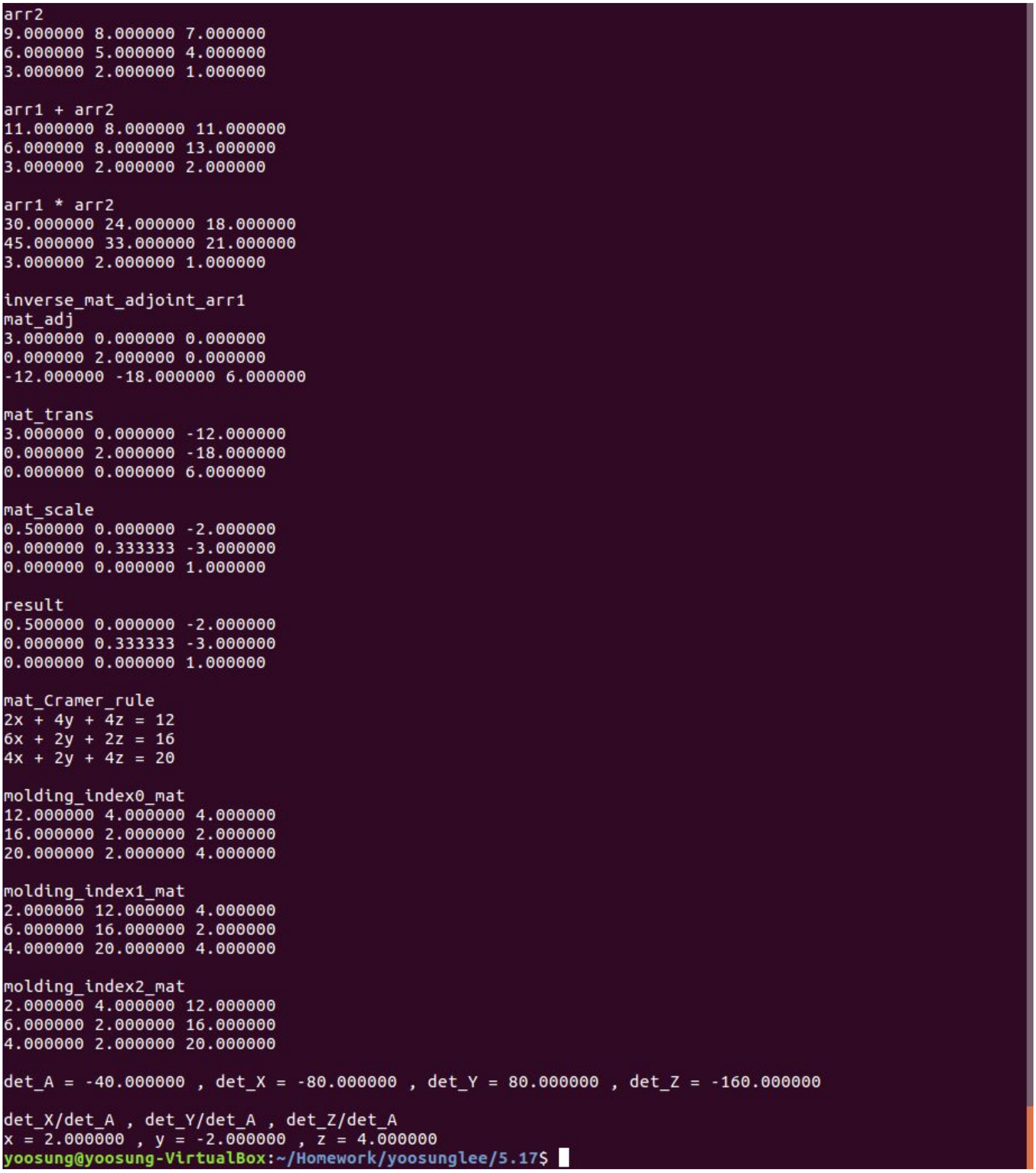
matrix.c

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


void mat_add(float (*ret)[3],float (*arr1)[3] ,float (*arr2)[3])
{
        int i,j ;

        for(i = 0 ; i < 3 ; i++){
        for(j = 0 ; j < 3; j++){
        ret[i][j] = arr1[i][j] + arr2[i][j] ;
        }
        }
}




void mat_print(float (*ret)[3])
{
        int i,j ;
        for(i= 0 ; i<3 ; i++){
        for( j= 0 ; j <3 ; j++){
        printf("%lf ",ret[i][j]);
        }
        printf("\n");
        }
        printf("\n");
}



void mat_mult(float (*ret)[3] , float (*arr1)[3] , float (*arr2)[3])
{

        ret[0][0] = arr1[0][0]*arr2[0][0]+arr1[0][1]*arr2[1][0]+arr1[0][2]*arr2[2][0];
        ret[0][1] = arr1[0][0]*arr2[0][1]+arr1[0][1]*arr2[1][1]+arr1[0][2]*arr2[2][1];
        ret[0][2] = arr1[0][0]*arr2[0][2]+arr1[0][1]*arr2[1][2]+arr1[0][2]*arr2[2][2];

        ret[1][0] = arr1[1][0]*arr2[0][0]+arr1[1][1]*arr2[1][0]+arr1[1][2]*arr2[2][0];
        ret[1][1] = arr1[1][0]*arr2[0][1]+arr1[1][1]*arr2[1][1]+arr1[1][2]*arr2[2][1];
        ret[1][2] = arr1[1][0]*arr2[0][2]+arr1[1][1]*arr2[1][2]+arr1[1][2]*arr2[2][2];

        ret[2][0] = arr1[2][0]*arr2[0][0]+arr1[2][1]*arr2[1][0]+arr1[2][2]*arr2[2][0];
        ret[2][1] = arr1[2][0]*arr2[0][1]+arr1[2][1]*arr2[1][1]+arr1[2][2]*arr2[2][1];
        ret[2][2] = arr1[2][0]*arr2[0][2]+arr1[2][1]*arr2[1][2]+arr1[2][2]*arr2[2][2];
/*
        int i,j;
        for( i = 0 ; i < 3 ; i ++){
        for(j = 0 ; j < 3 j++){
        ret[i][j] = arr1[i][0] * arr2[0][j] + arr1[i][1] * arr2[1][j] + arr1[i][2] * arr2[2][j];    }
        }
```

```
*/



/*

        int i,j,k,sum;

        for(i = 0 ; i < 3 ; i++){

        for(j = 0 ; j < 3; j++){

                sum = 0 ;

                for(k=0; k < 3; k++){

                sum+=arr1[i][k] * arr2[k][j];

                }

                ret[i][j] = sum;

        }

        }

*/



}

float func_det_A(float(*arr1)[3])
{
        return arr1[0][0] * (arr1[1][1] * arr1[2][2] - arr1[1][2] * arr1[2][1]) +
        arr1[0][1] * (arr1[1][2] * arr1[2][0] - arr1[1][0] * arr1[2][2]) +
        arr1[0][2] * (arr1[1][0] * arr1[2][1] - arr1[1][1] * arr1[2][0]);

}

void mat_trans(float (*ret)[3] , float (*arr1)[3])
{
        arr1[0][0] = ret[0][0];
        arr1[1][1] = ret[1][1];
        arr1[2][2] = ret[2][2];

        arr1[0][1] = ret[1][0];
        arr1[1][0] = ret[0][1];

        arr1[0][2] = ret[2][0];
        arr1[2][0] = ret[0][2];

        arr1[2][1] = ret[1][2];
        arr1[1][2] = ret[2][1];
```

```
}

void mat_scale(float det ,float (*arr1)[3] , float (*ret)[3])
{
        int i,j ;

        for(i = 0 ; i < 3; i++){
        for(j = 0 ; j < 3; j++){
        ret[i][j] = det * arr1[i][j] ;
        }
        }

        printf("mat_scale\n");
        mat_print(ret);
}


void mat_adj(float (*ret)[3] , float (*arr1)[3])
{

        ret[0][0] = arr1[1][1] * arr1[2][2] - arr1[1][2] * arr1[2][1];
        ret[0][1] = arr1[1][2] * arr1[2][0] - arr1[1][0] * arr1[2][2];
        ret[0][2] = arr1[1][0] * arr1[2][1] - arr1[1][1] * arr1[2][0];

        ret[1][0] = arr1[0][2] * arr1[2][1] - arr1[0][1] * arr1[2][2];
        ret[1][1] = arr1[0][0] * arr1[2][2] - arr1[0][2] * arr1[2][0];
        ret[1][2] = arr1[0][1] * arr1[2][0] - arr1[0][0] * arr1[2][0];

        ret[2][0] = arr1[0][1] * arr1[1][2] - arr1[0][2] * arr1[1][1];
        ret[2][1] = arr1[0][2] * arr1[1][0] - arr1[0][0] * arr1[1][2];
        ret[2][2] = arr1[0][0] * arr1[1][1] - arr1[0][1] * arr1[1][0];

   printf("mat_adj\n");
   mat_print(ret);

        mat_trans(ret,arr1);

   printf("mat_trans\n");
   mat_print(arr1);
}

// det -> adjoint ->transpose & scaling
int mat_inverse(float (*ret)[3] , float (*arr1)[3])
{
        int i,j;
        float det_A;

        det_A = func_det_A(arr1);

//      printf("det_A = %lf\n" ,det_A);

        if(det_A == 0){
        printf("역행렬 없음\n.");
        return 0;
        }
        mat_adj(ret ,arr1);

//      printf("1/det_A = %lf\n",1/det_A);

        mat_scale(1/det_A , arr1 , ret);

        return 1;


}



void molding_mat(float (*arr)[3], float *ans, int idx, float (*ret)[3])
{
        int i, j;

        for(i = 0; i < 3; i++)
        {
        for(j = 0; j < 3; j++)
        {
        if(j == idx)
        continue;
        ret[i][j] = arr[i][j];
        }

        ret[i][idx] = ans[i];
        }
}



void mat_Cramer_rule(float (*arr)[3] , float (*ret)[3] ,float *ans ,float *xyz)

{
  float det_A , det_X , det_Y , det_Z ;

  det_A = func_det_A(arr);

  printf("molding_index0_mat\n");
  molding_mat(arr ,ans ,0 ,ret);
  mat_print(ret);
  det_X = func_det_A(ret);

  printf("molding_index1_mat\n");
  molding_mat(arr ,ans ,1 ,ret);
  mat_print(ret);
  det_Y = func_det_A(ret);

  printf("molding_index2_mat\n");
  molding_mat(arr ,ans ,2 ,ret);
  mat_print(ret);
  det_Z = func_det_A(ret);

  printf("det_A = %lf , det_X = %lf , det_Y = %lf , det_Z = %lf \n",det_A ,det_X ,det_Y ,det_Z);

  xyz[0] = det_X / det_A ;
```

```
    xyz[1] = det_Y / det_A ;
    xyz[2] = det_Z / det_A ;


}




int main(void)
{
            int i;
            float ret[3][3] = {0};

            float arr[3][3] =  {{2 ,4 ,4},
                        {6 ,2 ,2},
                        {4 ,2 ,4}};


    //test_mat => arr1 & arr2

            float arr1[3][3] = {{2 ,0 ,4},
                        {0 ,3 ,9},
                        {0 ,0 ,1}};

            float arr2[3][3] = {{9 ,8 ,7},
                        {6 ,5 ,4},
                        {3 ,2 ,1}};

            float xyz[3] ={};
            float ans[3] ={12 ,16 ,20};


            printf("arr1\n");

            mat_print(arr1);
            printf("arr2\n");

            mat_print(arr2);
            printf("arr1 + arr2 \n");

            mat_add(ret ,arr1 ,arr2);
            mat_print(ret);

            printf("arr1 * arr2\n");
            mat_mult(ret ,arr1 ,arr2);
            mat_print(ret);

            printf("inverse_mat_adjoint_arr1\n");
            if(mat_inverse(ret,arr1)){
            printf("result\n");
    mat_print(ret);
            }

    printf("mat_Cramer_rule \n2x + 4y + 4z = 12 \n6x + 2y + 2z = 16 \n4x + 2y + 4z = 20 \n\n");
    mat_Cramer_rule(arr ,ret ,ans ,xyz);
    printf("\ndet_X/det_A , det_Y/det_A , det_Z/det_A \nx = %lf , y = %lf , z = %lf
\n",xyz[0],xyz[1],xyz[2]);


        return 0;
}
```



```
arr2
9.000000 8.000000 7.000000
6.000000 5.000000 4.000000
3.000000 2.000000 1.000000

arr1 + arr2
11.000000 8.000000 11.000000
6.000000 8.000000 13.000000
3.000000 2.000000 2.000000

arr1 * arr2
30.000000 24.000000 18.000000
45.000000 33.000000 21.000000
3.000000 2.000000 1.000000

inverse_mat_adjoint_arr1
mat_adj
3.000000 0.000000 0.000000
0.000000 2.000000 0.000000
-12.000000 -18.000000 6.000000

mat_trans
3.000000 0.000000 -12.000000
0.000000 2.000000 -18.000000
0.000000 0.000000 6.000000

mat_scale
0.500000 0.000000 -2.000000
0.000000 0.333333 -3.000000
0.000000 0.000000 1.000000

result
0.500000 0.000000 -2.000000
0.000000 0.333333 -3.000000
0.000000 0.000000 1.000000

mat_Cramer_rule
2x + 4y + 4z = 12
6x + 2y + 2z = 16
4x + 2y + 4z = 20

molding_index0_mat
12.000000 4.000000 4.000000
16.000000 2.000000 2.000000
20.000000 2.000000 4.000000

molding_index1_mat
2.000000 12.000000 4.000000
6.000000 16.000000 2.000000
4.000000 20.000000 4.000000

molding_index2_mat
2.000000 4.000000 12.000000
6.000000 2.000000 16.000000
4.000000 2.000000 20.000000

det_A = -40.000000 , det_X = -80.000000 , det_Y = 80.000000 , det_Z = -160.000000

det_X/det_A , det_Y/det_A , det_Z/det_A
x = 2.000000 , y = -2.000000 , z = 4.000000
yoosung@yoosung-VirtualBox:~/Homework/yoosunglee/5.17$ 
```