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

#56

2018.05.18

강사:Innova Lee(이 상훈)

학생: 김시윤

과제

행렬 소스코드 작성

나의 소스코드.

Hangryul.h

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

typedef struct hangryul hang;

struct hangryul{

        float x;
        float y;
        float z;

        void (* add)(hang *, hang *, hang *);
        void (* sub)(hang *, hang *, hang *);
    void (* mult)(hang *, hang *, hang *);
        float (* det)(hang *);
        void (* crammer)(hang *,hang *,float *,hang);
        void (* adj)(hang *, hang *);
        void (* trans)(hang *);
        void (* scale)(hang *, float );
        void (* adj_invert)(hang *,hang );
        void (* print)(hang *);
};
```

```
void hang_add(hang *a, hang *b, hang *r)
{
        int i;

        for(i=0; i<3; i++)
        {
                r[i].x = a[i].x + b[i].x;
                r[i].y = a[i].y + b[i].y;
                r[i].z = a[i].z + b[i].z;
        }
}

void hang_sub(hang *a, hang *b, hang *r)
{
    int i;

    for(i=0; i<3; i++)
    {
        r[i].x = a[i].x - b[i].x;
        r[i].y = a[i].y - b[i].y;
        r[i].z = a[i].z - b[i].z;
    }

}

void hang_mult(hang *a, hang *b, hang *r)
{
        r[0].x = a[0].x * b[0].x + a[0].y * b[1].x + a[0].z * b[2].x;
        r[0].y = a[0].x * b[0].y + a[0].y * b[1].y + a[0].z * b[2].y;
        r[0].z = a[0].x * b[0].z + a[0].y * b[1].z + a[0].z * b[2].z;

    r[1].x = a[1].x * b[0].x + a[1].y * b[1].x + a[1].z * b[2].x;
    r[1].y = a[1].x * b[0].y + a[1].y * b[1].y + a[1].z * b[2].y;
```

```
      r[1].z = a[1].x * b[0].z + a[1].y * b[1].z + a[1].z * b[2].z;

      r[2].x = a[2].x * b[0].x + a[2].y * b[1].x + a[2].z * b[2].x;
      r[2].y = a[2].x * b[0].y + a[2].y * b[1].y + a[2].z * b[2].y;
      r[2].z = a[2].x * b[0].z + a[2].y * b[1].z + a[2].z * b[2].z;
}

float hang_det(hang *a)
{
           return  a[0].x * ((a[1].y * a[2].z) - (a[1].z*a[2].y)) + (-
(a[0].y*((a[1].x*a[2].z) - (a[1].z*a[2].x)))) + (a[0].z*((a[1].x*a[2].y) -
(a[1].y*a[2].x)));
}

void hang_crammer(hang a[3],hang *c,float *r2,hang r)
{
           int i;
           float res, res2;

           hang x[3],y[3],z[3];

           for(i = 0; i<3;i++)
           {
                      x[i].x = c[i].x;
                      x[i].y = a[i].y;
                      x[i].z = a[i].z;

                  y[i].y = c[i].x;
           y[i].x = a[i].x;
           y[i].z = a[i].z;

                  z[i].z = c[i].x;
           z[i].x = a[i].x;
            z[i].y = a[i].y;
           }
           res = r.det(a);
           res2 = r.det(x);
           r2[0] = res2/res;

           res = r.det(a);
           res2 = r.det(y);
           r2[1] = res2/res;

           res = r.det(a);
           res2 = r.det(z);
           r2[2] = res2/res;
}

void hang_adj(hang a[3],hang *r)
{
           r[0].x = (a[1].y*a[2].z) - (a[1].z*a[2].y);
           r[0].y = (a[1].z*a[2].x) - (a[1].x*a[2].z);
           r[0].z = (a[1].x*a[2].y) - (a[1].y*a[2].x);

           r[1].x = (a[0].z*a[2].y) - (a[0].y*a[2].z);
           r[1].y = (a[0].x*a[2].z) - (a[0].z*a[2].x);
           r[1].z = (a[0].y*a[2].x) - (a[0].x*a[2].y);

           r[2].x = (a[0].y*a[1].z) - (a[0].z*a[1].y);
           r[2].y = (a[0].z*a[1].x) - (a[0].x*a[1].z);
           r[2].z = (a[0].x*a[1].y) - (a[0].y*a[1].x);
}

void hang_scale(hang *r, float n)
{
           int i;
```

```
        for(i=0; i<3; i++)
        {
                r[i].x = r[i].x * n;
                r[i].y = r[i].y * n;
                r[i].z = r[i].z * n;
        }
}

void hang_trans(hang a[3])
{
        float tmp;

        tmp = a[0].y;
        a[0].y = a[1].x;
        a[1].x = tmp;

        tmp = a[0].z;
        a[0].z = a[2].x;
        a[2].x = tmp;

        tmp = a[1].z;
        a[1].z = a[2].y;
        a[2].y = tmp;
}

void hang_adj_invert(hang a[3],hang r )
{
        hang res[3] ={0};
        float det1;
        int i;

        det1 = 1.0/r.det(a);
        r.adj(a,res);
        r.scale(res,det1);
```

```
        r.trans(a);
        for(i=0;i<3;i++)
        {
                a[i].x = res[i].x;
                a[i].y = res[i].y;
                a[i].z = res[i].z;
        }
}

void print_hang(hang *r)
{
        int i;
        for(i=0;i<3;i++)
    printf(" %f,  %f,  %f\n", r[i].x, r[i].y, r[i].z);
}
```

Hangryul.c

```
#include "hangryul.h"
#include <stdio.h>

int main(void)
{
        hang A[3] ={{2,4,4},{6,2,2},{4,2,4}};
        hang B[3] ={{1,0,0},{0,1,0},{0,0,1}};
        hang C[3] ={{12},{16},{20}};
        hang D[3] ={{2,0,4},{0,3,9},{0,0,1}};
        hang R2[3] = {0};
        float R3[3] = {0};
        hang R ={0,0,0,hang_add ,hang_sub
,hang_mult,hang_det,hang_crammer,hang_adj,hang_trans,hang_scale,hang_adj_invert,print_hang};
```

```
	float res=0;

	printf("hangryul A\n");

	R.print(A);

	printf("hangryul B\n");
	R.print(B);

	printf("hangryul A+B\n");
	R.add(A,B,R2);
	R.print(R2);

	printf("hangryul A-B\n");
	R.sub(A,B,R2);
	R.print(R2);

	printf("mult A*B \n");
	R.mult(A,B,R2);
	R.print(R2);

	printf("det A\n");
	printf("%lf\n",R.det(A));

	R.print(C);

	printf("crammer A\n");
	R.crammer(A,C,R3,R);
	printf("x = %f , y = %f , z = %f \n",R3[0],R3[1],R3[2]);

	printf("adj \n");
	R.adj(A,R2);
	R.print(R2);
	printf("trans \n");
```

```
	R.trans(R2);
	R.print(R2);

	printf("%f = Ddet \n",R.det(D));

	printf("adj D\n");
	R.adj(D,R2);
	R.print(R2);

	printf("adj_invert D\n");
	R.adj_invert(D,R);
	R.print(D);

return 0;
}
```

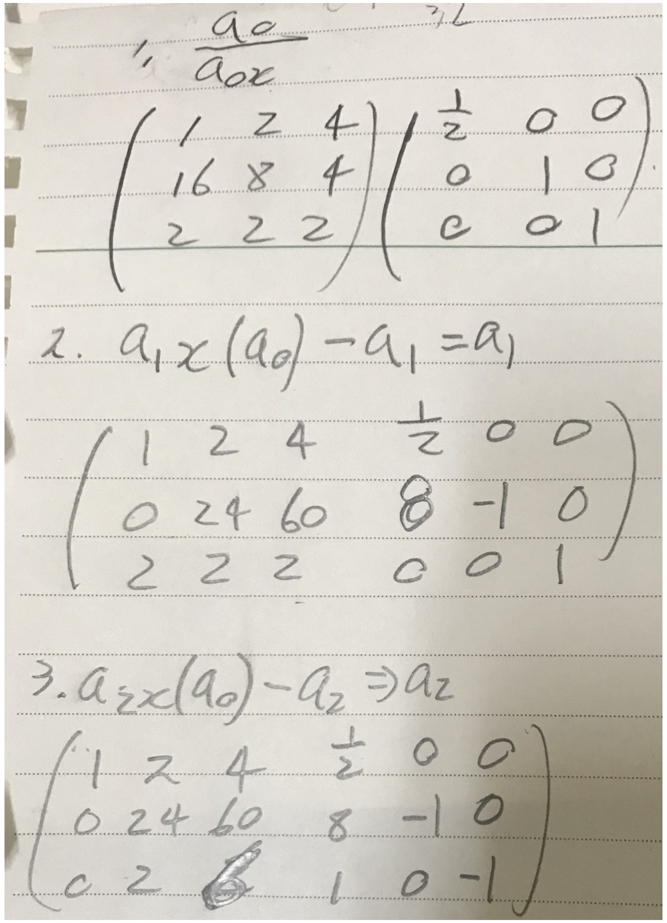
--- Adj 역행렬까지 성공

가우스 역행렬과 연립방정식.

역행렬을 구하면 연립도 바로 가능하므로 오늘은 소스 코드 짜기전 알고리즘을 수식화 하였다.

밤이 깊어 수식화한 알고리즘을 소스코드로 구현하지 못하였다..

여기서 문제 !!
여러 상황에 따른 조건문을 추가 해주어야 한다.



1. $\frac{a_0}{a_0 x}$

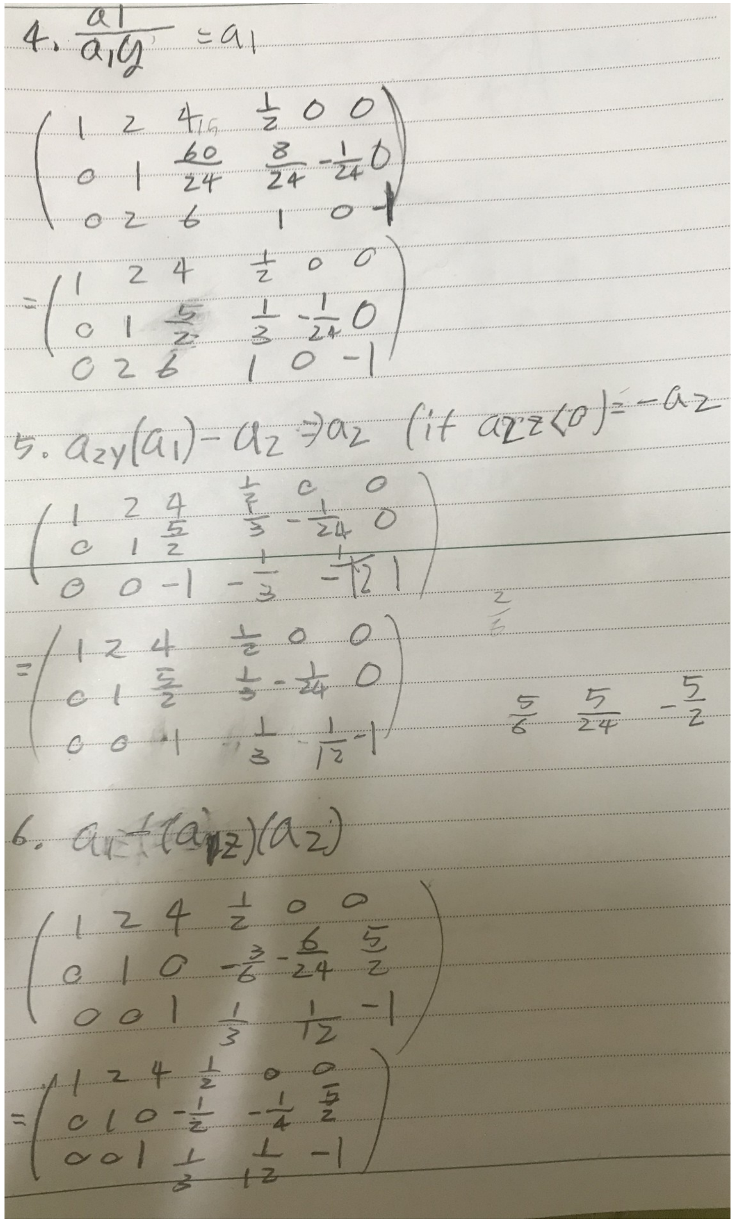
$$\begin{pmatrix} 1 & 2 & 4 \\ 16 & 8 & 4 \\ 2 & 2 & 2 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. $a_1 x (a_0) - a_1 = a_1$

$$\begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 24 & 60 & 8 & -1 & 0 \\ 2 & 2 & 2 & 0 & 0 & 1 \end{pmatrix}$$

3. $a_2 x (a_0) - a_2 \Rightarrow a_2$

$$\begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 24 & 60 & 8 & -1 & 0 \\ 0 & 2 & 6 & 1 & 0 & -1 \end{pmatrix}$$



4. $\frac{a1}{a_1 y} \doteq a_1$

$$\begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{60}{24} & \frac{8}{24} & -\frac{1}{24} & 0 \\ 0 & 2 & 6 & 1 & 0 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{5}{2} & \frac{1}{3} & -\frac{1}{24} & 0 \\ 0 & 2 & 6 & 1 & 0 & -1 \end{pmatrix}$$

5. $a_{2y}(a_1) - a_2 \Rightarrow a_2$ (if $a_{22} < 0$) $= -a_2$

$$\begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{5}{2} & \frac{1}{3} & -\frac{1}{24} & 0 \\ 0 & 0 & -1 & -\frac{1}{3} & -\frac{1}{12} & 1 \end{pmatrix}$$

$\frac{2}{3}$

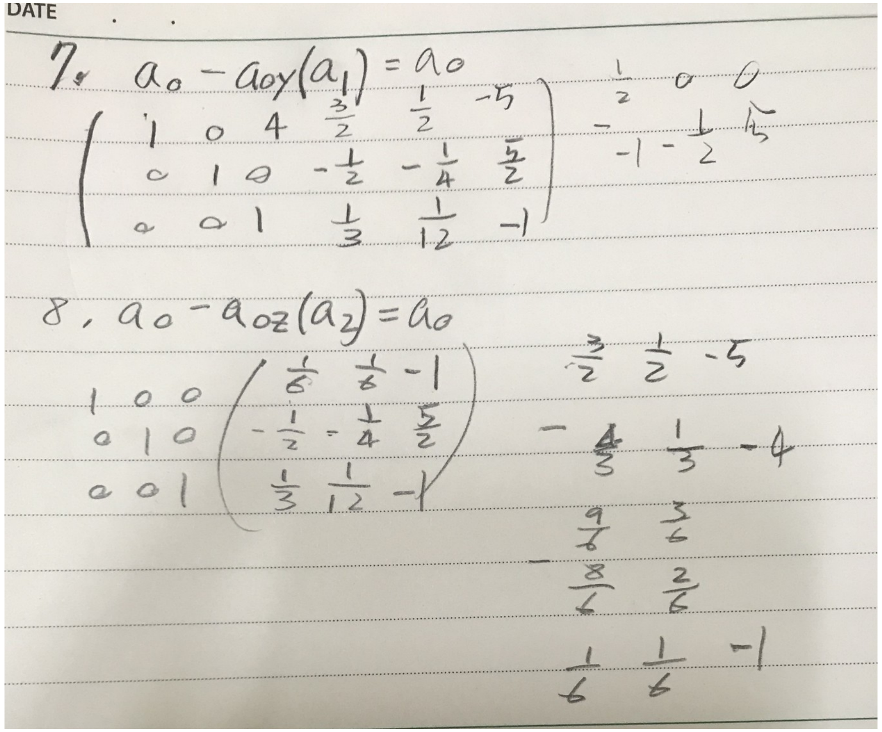
$$= \begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & \frac{5}{2} & \frac{1}{3} & -\frac{1}{24} & 0 \\ 0 & 0 & 1 & \frac{1}{3} & \frac{1}{12} & -1 \end{pmatrix}$$

$\frac{5}{6} \quad \frac{5}{24} \quad -\frac{5}{2}$

6. $a_1 - (a_{1z})(a_2)$

$$\begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & -\frac{3}{6} & -\frac{6}{24} & \frac{5}{2} \\ 0 & 0 & 1 & \frac{1}{3} & \frac{1}{12} & -1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 4 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & -\frac{1}{2} & -\frac{1}{4} & \frac{5}{2} \\ 0 & 0 & 1 & \frac{1}{3} & \frac{1}{12} & -1 \end{pmatrix}$$



DATE . .

7. $a_0 - a_{0y}(a_1) = a_0$

$$\begin{pmatrix} 1 & 0 & 4 & \frac{3}{2} & \frac{1}{2} & -5 \\ 0 & 1 & 0 & -\frac{1}{2} & -\frac{1}{4} & \frac{5}{2} \\ 0 & 0 & 1 & \frac{1}{3} & \frac{1}{12} & -1 \end{pmatrix}$$

$\frac{1}{2} \quad 0 \quad 0$

$-$

$-1 \quad -\frac{1}{2} \quad 5$

8. $a_0 - a_{0z}(a_2) = a_0$

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & -1 \\ -\frac{1}{2} & -\frac{1}{4} & \frac{5}{2} \\ \frac{1}{3} & \frac{1}{12} & -1 \end{pmatrix}$$

$\frac{3}{2} \quad \frac{1}{2} \quad -5$

$- \quad \frac{4}{3} \quad \frac{1}{3} \quad -4$

$\frac{9}{6} \quad \frac{3}{6}$

$- \quad \frac{8}{6} \quad \frac{2}{6}$

$\frac{1}{6} \quad \frac{1}{6} \quad -1$