Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정 #57

강사:Innova Lee(이 상훈)
학생: 김시윤

과제

가우스 소거법 역행렬 구현.

```
#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 (* gaus_invert)(hang *,hang *,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(res);
		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 scale2(float (*t)[6],int raw,float n)
{
		int i;
		int j;

		for(i=0; i<6; i++)
		{
			t[raw][i] = t[raw][i] * n;
		}


}

void copy_hang(float (*t)[6], float (*c)[6])
{
		int i,j;

		for(i = 0; i < 3; i++)
		{
			for(j=0; j<6; j++)
			{
				c[i][j] = t[i][j];
			}
		}
}

void sub2(float (*t)[6],int a, int b)
{
		int i;
```

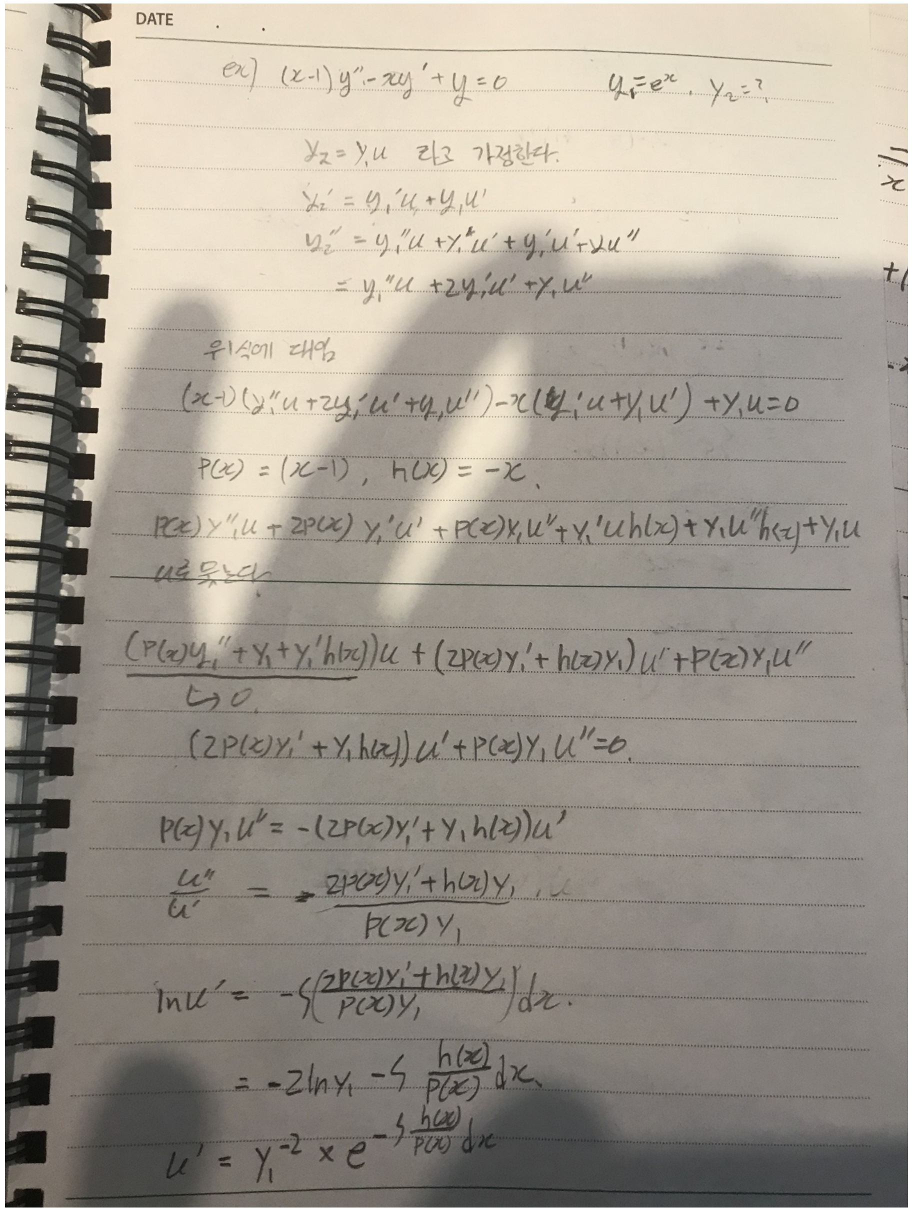
```
	for(i = 0; i < 6; i++)
	{
	t[b][i] = t[a][i] - t[b][i];
	}
}

void compare_hang(float (*tmp)[6],int n)
{
	int i;

	if(tmp[n][n] < 0)
	{
		for(i=0; i<6; i++)
		{
			tmp[n][i] = (-1.0)*tmp[n][i];
		}
	}

}

void replace_hang(float (*tmp)[6], float (*ret)[6],int n)
{
	int i;

	for(i=0;i<6;i++)
	{
		tmp[n][i] = ret[n][i];
	}

}

void hang_gaus_invert(hang *a,hang *d,hang *re,hang r)
{
	float tmp[3][6] = {0};
	float copy[3][6] = {0};
	float ret[3][6] = {0};
	float data;
	int i,j;

	for(i=0; i<3; i++)
	{
		tmp[i][0] = a[i].x;
		tmp[i][1] = a[i].y;
		tmp[i][2] = a[i].z;

		tmp[i][3] = d[i].x;
		tmp[i][4] = d[i].y;
		tmp[i][5] = d[i].z;
	}

	copy_hang(tmp,copy);

	data=(1.0/tmp[0][0]);
	scale2(tmp,0,data);
	copy_hang(tmp,ret);
	data = tmp[1][0];
	scale2(tmp,0,data);

	sub2(tmp,0,1);

	replace_hang(tmp,ret,0);
	copy_hang(tmp,ret);
	data = tmp[2][0];
	scale2(tmp,0,data);
	sub2(tmp,0,2);

	replace_hang(tmp,ret,0);
	copy_hang(tmp,ret);
	data = 1.0/tmp[1][1];
	scale2(tmp,1,data);

	compare_hang(tmp,1);

	copy_hang(tmp,ret);

	data = tmp[2][1];
	scale2(tmp,1,data);
	sub2(tmp,1,2);

	replace_hang(tmp,ret,1);

	data = 1.0/tmp[2][2];
	scale2(tmp,2,data);

	compare_hang(tmp,2);

	copy_hang(tmp,ret);

	data = tmp[1][2];
	scale2(tmp,2,data);
	sub2(tmp,2,1);

	replace_hang(tmp,ret,2);

	compare_hang(tmp,1);

	copy_hang(tmp,ret);

	data = tmp[0][1];
	scale2(tmp,1,data);
	sub2(tmp,1,0);
```

```
	replace_hang(tmp,ret,1);
	compare_hang(tmp,0);

	copy_hang(tmp,ret);

	data = tmp[0][2];
	scale2(tmp,2,data);
	sub2(tmp,2,0);

	replace_hang(tmp,ret,2);
	compare_hang(tmp,0);

	for(i = 0; i<3; i++)
	{
		a[i].x = tmp[i][0];
		a[i].y = tmp[i][1];
		a[i].z = tmp[i][2];
		re[i].x = tmp[i][3];
		re[i].y = tmp[i][4];
		re[i].z = tmp[i][5];
	}
}

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);
}
```

```c
#include "hangryul2.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 B2[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 D2[3] ={{2,0,4},{0,3,9},{0,0,1}};
	hang G[3] = {{2,4,8},{16,8,4},{2,2,2}};
	hang G2[3] = {{2,4,8},{16,8,4},{2,2,2}};
	hang R2[3] = {0};
	hang R4[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,han
g_adj_invert,hang_gaus_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);
	printf("\n");
	printf("adj_invert * origin\n");
	R.mult(D,D2,R4);
```

```
        R.print(R4);



        printf("gaus_invert A\n");
        R.gaus_invert(G,B,R2,R);
        printf("gaus[0~2]\n");
        R.print(G);

        printf("gaus[3~5]\n");
        R.print(R2);

        printf("origin * gaus_invert\n");
        R.mult(G2,R2,R4);
        R.print(R4);

        printf("gaus_invert ADJ\n");
        R.gaus_invert(D2,B2,R2,R);

        printf("gaus[0~2]\n");
        R.print(D2);
        printf("gaus[3~5]\n");
        R.print(R2);


return 0;
}
```

미분 기본 개념.



DATE . .

ex) $(x-1)y'' - xy' + y = 0$ $\qquad y_1 = e^x,\ y_2 = ?$

$y_2 = y_1 u$ 라고 가정한다.

$y_2' = y_1' u + y_1 u'$

$y_2'' = y_1'' u + y_1' u' + y_1' u' + y_1 u''$

$= y_1'' u + 2y_1' u' + y_1 u''$

위식에 대입

$(x-1)(y_1'' u + 2y_1' u' + y_1 u'') - x(y_1' u + y_1 u') + y_1 u = 0$

$P(x) = (x-1),\ h(x) = -x.$

$P(x) y_1'' u + 2P(x) y_1' u' + P(x) y_1 u'' + y_1' u h(x) + y_1 u'' h(x) + y_1 u$

~~u로 묶는다.~~

$\underbrace{(P(x) y_1'' + y_1 + y_1' h(x))}_{\to 0} u + (2P(x) y_1' + h(x) y_1) u' + P(x) y_1 u''$

$(2P(x) y_1' + y_1 h(x)) u' + P(x) y_1 u'' = 0.$

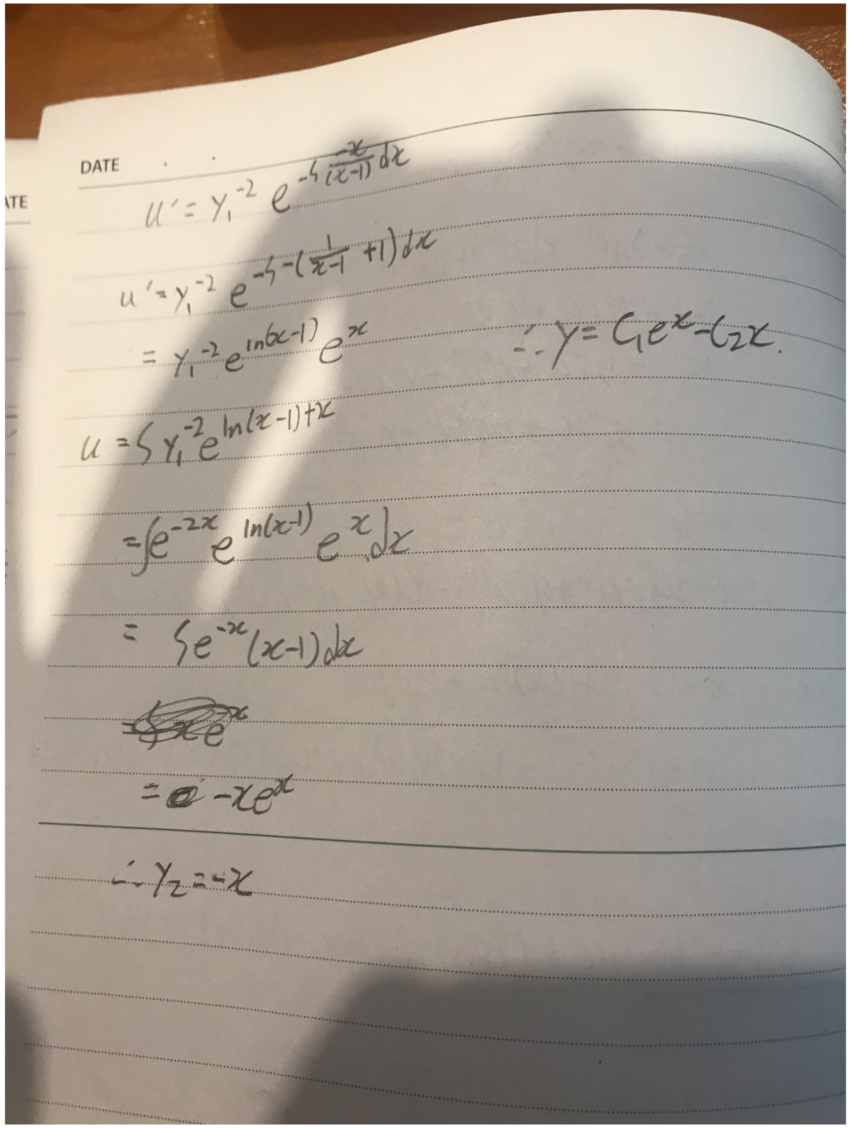
$P(x) y_1 u'' = -(2P(x) y_1' + y_1 h(x)) u'$

$\frac{u''}{u'} = -\frac{2P(x) y_1' + h(x) y_1}{P(x) y_1}$

$\ln u' = -\int \left(\frac{2P(x) y_1' + h(x) y_1}{P(x) y_1}\right) dx.$

$= -2\ln y_1 - \int \frac{h(x)}{P(x)} dx.$

$u' = y_1^{-2} \times e^{-\int \frac{h(x)}{P(x)} dx}$



DATE . .

$u' = y_1^{-2} e^{-\int \frac{-x}{(x-1)} dx}$

$u' = y_1^{-2} e^{-\int -(\frac{1}{x-1} + 1) dx}$

$= y_1^{-2} e^{\ln(x-1)} e^{x}$ $\qquad \therefore y = C_1 e^x - C_2 x.$

$u = \int y_1^{-2} e^{\ln(x-1) + x}$

$= \int e^{-2x} e^{\ln(x-1)} e^{x} dx$

$= \int e^{-x}(x-1) dx$

~~$= \int x e^{-x}$~~

$= -xe^{-x}$

$\therefore y_2 = -x$