TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

목차

6. 수학

6. 수학 – 3 X 3 역행렬

$$A^{-1}=\left(\frac{1}{\det A}\right)\operatorname{adj}A$$

adjoint 는 아래와 같이 정의합니다.

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \text{일 때}$$

$$\operatorname{adj} A=\begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{pmatrix}^T=\begin{pmatrix} C_{11} & C_{21} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{n2} \\ \vdots & \vdots & & \vdots \\ C_{1n} & C_{2n} & \cdots & C_{nn} \end{pmatrix}$$

determinant 는 아래와 같이 정의됩니다.

$$\det A=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}=a_{i1}C_{i1}+a_{i2}C_{i2}+\cdots+a_{in}C_{in}=a_{1j}C_{1j}+a_{2j}C_{2j}+\cdots+a_{nj}C_{nj}$$

$$A=\begin{pmatrix} 3 & 1 & 4 \\ 2 & 3 & 1 \\ 1 & 2 & 5 \end{pmatrix}$$

$$\det A=\begin{vmatrix} 3 & 1 & 4 \\ 2 & 3 & 1 \\ 1 & 2 & 5 \end{vmatrix}=3\times\begin{vmatrix} 3 & 1 \\ 2 & 5 \end{vmatrix}-2\times\begin{vmatrix} 1 & 4 \\ 2 & 5 \end{vmatrix}+1\times\begin{vmatrix} 1 & 4 \\ 3 & 1 \end{vmatrix}=39+6-11=34$$

$$\operatorname{adj}A=\begin{pmatrix} +\begin{vmatrix} 3 & 1 \\ 2 & 5 \end{vmatrix} & -\begin{vmatrix} 2 & 1 \\ 1 & 5 \end{vmatrix} & +\begin{vmatrix} 2 & 3 \\ 1 & 2 \end{vmatrix} \\ -\begin{vmatrix} 1 & 4 \\ 2 & 5 \end{vmatrix} & +\begin{vmatrix} 3 & 4 \\ 1 & 5 \end{vmatrix} & -\begin{vmatrix} 3 & 1 \\ 1 & 2 \end{vmatrix} \\ +\begin{vmatrix} 1 & 4 \\ 3 & 1 \end{vmatrix} & -\begin{vmatrix} 3 & 4 \\ 2 & 1 \end{vmatrix} & +\begin{vmatrix} 3 & 1 \\ 2 & 3 \end{vmatrix} \end{pmatrix}^T=\begin{pmatrix} 13 & -9 & 1 \\ 3 & 11 & -5 \\ -11 & 5 & 7 \end{pmatrix}^T=\begin{pmatrix} 13 & 3 & -11 \\ -9 & 11 & 5 \\ 1 & -5 & 7 \end{pmatrix}$$

$$\therefore A^{-1}=\frac{1}{34}\begin{pmatrix} 13 & 3 & -11 \\ -9 & 11 & 5 \\ 1 & -5 & 7 \end{pmatrix}$$

6. 수학 – 가우스 소거법 역행렬

이 가우스 소거법을 이용하여 역행렬을 구한다.

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

의 역행렬을 A^{-1}라고 하면 아래와 같이 곱해서 단위행렬 I가 되는 행렬이다.

$$AA^{-1} = A^{-1}A = I$$

$$AA^{-1} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$[A|I] = \left[\begin{array}{ccc|ccc} 2 & -1 & 0 & 1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 2 & 0 & 0 & 1 \end{array}\right]$$

$$\boxed{[I|A^{-1}] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} \end{array}\right]}$$

(1) $\begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 1 \\ 1 & -2 & 3 \end{pmatrix} \Bigg| \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

$\begin{pmatrix} 1 & -1 & 1 \\ 0 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix} \Bigg| \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \Leftarrow \begin{cases} (2\text{행})+(1\text{행})\times(-2) \\ (3\text{행})+(1\text{행})\times(-1) \end{cases}$

$\begin{pmatrix} 1 & 0 & \frac{2}{3} \\ 0 & 1 & -\frac{1}{3} \\ 0 & 0 & \frac{5}{3} \end{pmatrix} \Bigg| \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 \\ -\frac{2}{3} & \frac{1}{3} & 0 \\ -\frac{5}{3} & \frac{1}{3} & 1 \end{pmatrix} \Leftarrow \begin{cases} (1\text{행})+(2\text{행})\times\frac{1}{3} \\ (2\text{행})\times\frac{1}{3} \\ (3\text{행})+(2\text{행})\times\frac{1}{3} \end{cases}$

$\begin{pmatrix} 1 & 0 & \frac{2}{3} \\ 0 & 1 & -\frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix} \Bigg| \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 \\ -\frac{2}{3} & \frac{1}{3} & 0 \\ -1 & \frac{1}{5} & \frac{3}{5} \end{pmatrix} \Leftarrow (3\text{행})\times\frac{3}{5}$

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \Bigg| \begin{pmatrix} 1 & \frac{1}{5} & -\frac{2}{5} \\ -1 & \frac{2}{5} & \frac{1}{5} \\ -1 & \frac{1}{5} & \frac{3}{5} \end{pmatrix} \Leftarrow \begin{cases} (1\text{행})+(3\text{행})\times\left(-\frac{2}{3}\right) \\ (2\text{행})+(3\text{행})\times\frac{1}{3} \end{cases}$

$\begin{pmatrix} 1 & \frac{1}{5} & -\frac{2}{5} \\ -1 & \frac{2}{5} & \frac{1}{5} \\ -1 & \frac{1}{5} & \frac{3}{5} \end{pmatrix} \begin{pmatrix} 9 \\ 5 \\ 20 \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 4 \end{pmatrix} \quad \therefore \begin{cases} x=2 \\ y=-3 \\ z=4 \end{cases}$

6. 수학 – 크래머 공식

다음 연립일차방정식을 크래머 공식을 이용하여 해를 구하시오.

$$\begin{cases} x+2z=6 \\ 3x-4y-6z=-30 \\ x+2y-3z=-8 \end{cases}$$

풀이)

연립일차방정식의 계수행렬은 다음과 같다.

$$A=\begin{pmatrix} 1 & 0 & 2 \\ 3 & -4 & -6 \\ 1 & 2 & -3 \end{pmatrix},\ B=\begin{pmatrix} 6 \\ -30 \\ -8 \end{pmatrix}$$

$|A|=1\cdot(24)-0\cdot(-3)+2(10)=44$ (1행에 대한 여인수전개)

$$A_1=\begin{pmatrix} 6 & 0 & 2 \\ -30 & -4 & -6 \\ -8 & 2 & -3 \end{pmatrix} \Rightarrow |A_1|=6\cdot(24)+2(-92)=-40$$

$$A_2=\begin{pmatrix} 1 & 6 & 2 \\ 3 & -30 & -6 \\ 1 & -8 & -3 \end{pmatrix} \Rightarrow |A_2|=1\cdot(42)-6\cdot(-3)+2(6)=72$$

$$A_3=\begin{pmatrix} 1 & 0 & 6 \\ 3 & -4 & -30 \\ 1 & 2 & -8 \end{pmatrix} \Rightarrow |A_3|=1\cdot(92)+6\cdot(10)=152$$

$$\Rightarrow x=\frac{|A_1|}{|A|}=\frac{-40}{44}=-\frac{10}{11}$$

$$y=\frac{|A_2|}{|A|}=\frac{72}{44}=\frac{18}{11}$$

$$z=\frac{|A_3|}{|A|}=\frac{152}{44}=\frac{38}{11}$$

$\therefore\ x=-\frac{10}{11},\quad y=\frac{18}{11},\quad z=\frac{38}{11}$

6. 수학 – matrix 3X3 프로그래밍 1

< gauss.c >

```c
#include "gauss.h"


int main(void){

    max3 A = { {2, 0, 0}, {0, 3, 0}, {4, 9, 1} };
    max3 B = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
    max3 C = { {2, 4, 8}, {16, 8, 4}, {2, 2, 2} };
    max3 R = { {}, {}, {}, max3_print, max3_add, max3_sub, max3_mul,
                  max3_det, max3_inverse, max3_gauss_eli };

    float det;

    R.add(A,B, &R);
    printf("ADD\n");
    R.print(R);

    R.sub(A,B, &R);
    printf("SUB\n");
    R.print(R);

    R.mul( 3.0, A, &R);
    printf("MUL\n");
    R.print(R);

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

    R.inverse(A, &R);
    R.print(R);

    R.gauss(C, &R);
//  R.print(R);

    return 0;
}
```

< gauss.h >

```c
#ifndef __MATRIX_3D_H__
#define __MATRIX_3D_H__

#include "vector.h"
#include <stdio.h>
#include <math.h>

typedef struct matrix_3d max3;

struct matrix_3d{

    float x[3];
    float y[3];
    float z[3];

    void (* print)(max3);
    void (* add)(max3, max3, max3 *);
    void (* sub)(max3, max3, max3 *);
    void (* mul)(float, max3, max3 *);
    float (* det)(max3);

    void (* inverse)(max3, max3 *);
    void (* gauss)(max3, max3 *);
    void (* crammer)(max3, max3, max3 *);
};

void max3_print(max3 a){
    int i;
    for(i=0;i<3; i++){
        printf("\t%lf\t%lf\t%lf\n", a.x[i], a.y[i], a.z[i]);
    }
}


void max3_add(max3 a, max3 b, max3 *r){

    int i;

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

```c
void max3_sub(max3 a, max3 b, max3 *r){
    int i;
    for( i=0; i<3;i++){
        r->x[i] = a.x[i] - b.x[i];
        r->y[i] = a.y[i] - b.y[i];
        r->z[i] = a.z[i] - b.z[i];
    }
}

void max3_mul(float num, max3 a, max3 *r){
    int i;
    for( i=0; i<3;i++){
        r->x[i] = num * a.x[i];
        r->y[i] = num * a.y[i];
        r->z[i] = num * a.z[i];
    }

}

float max3_det(max3 a){

    float w0 = a.x[0] * ( a.y[1] *a.z[2] - a.z[1] * a.y[2]);
    float w1 = -a.y[0] * ( a.x[1] *a.z[2] - a.z[1] * a.x[2]);
    float w2 = a.z[0] * ( a.x[1] *a.y[2] - a.y[1] * a.x[2]);

    float det = w0 + w1 + w2;
    return det;
}
```

6. 수학 – matrix 3X3 프로그래밍 1

< gauss.h >

```c
void max3_inverse(max3 a, max3 *r){

    float det = max3_det(a);
    max3 adj;
    max3 t_pose;

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

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

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

    printf("ADJ\n");
    max3_print(adj);

    t_pose.x[0] = adj.x[0];
    t_pose.x[1] = adj.y[0];
    t_pose.x[2] = adj.z[0];

    t_pose.y[0] = adj.x[1];
    t_pose.y[1] = adj.y[1];
    t_pose.y[2] = adj.z[1];

    t_pose.z[0] = adj.x[2];
    t_pose.z[1] = adj.y[2];
    t_pose.z[2] = adj.z[2];

    printf("TRANSPOSE\n");
    max3_print(t_pose);
    printf("INVERSE\n");
    max3_mul( 1.0/det, t_pose, r);

}
```

```c
void max3_gauss_eli(max3 a, max3 *r){

    int i=0;
    float coef;
    vec3 e[3] = { {1, 0, 0}, {0, 1, 0}, {0, 0, 1} };
    vec3 g[3] ={{a.x[0], a.y[0], a.z[0]}, {a.x[1], a.y[1], a.z[1]}
              , {a.x[2], a.y[2], a.z[2]}   };

//  vec3 g[3] = { {2,4,8}, {16, 8, 4}, {2, 2 , 2} };

    /*앞에 두자리*/
    coef = g[1].x / g[0].x;
    vec3_scale( coef, g[0], &g[0]);
    vec3_scale( coef, e[0], &e[0]);
    vec3_sub(g[1], g[0], &g[1]);
    vec3_sub(e[1], e[0], &e[1]);

    coef = g[2].x / g[0].x;
    vec3_scale( coef, g[0], &g[0]);
    vec3_scale( coef, e[0], &e[0]);
    vec3_sub(g[2], g[0], &g[2]);
    vec3_sub(e[2], e[0], &e[2]);


    /*중간 두자리*/
    coef = g[0].y / g[1].y;
    vec3_scale( coef, g[1], &g[1]);
    vec3_scale( coef, e[1], &e[1]);
    vec3_sub(g[0], g[1], &g[0]);
    vec3_sub(e[0], e[1], &e[0]);

    coef = g[2].y / g[1].y;
    vec3_scale( coef, g[1], &g[1]);
    vec3_scale( coef, e[1], &e[1]);
    vec3_sub(g[2], g[1], &g[2]);
    vec3_sub(e[2], e[1], &e[2]);

    /*마지막 두자리*/
    coef = g[0].z / g[2].z;
    vec3_scale( coef, g[2], &g[2]);
    vec3_scale( coef, e[2], &e[2]);
    vec3_sub(g[0], g[2], &g[0]);
    vec3_sub(e[0], e[2], &e[0]);

    coef = g[1].z / g[2].z;
    vec3_scale( coef, g[2], &g[2]);
    vec3_scale( coef, e[2], &e[2]);
    vec3_sub(g[2], g[1], &g[1]);
    vec3_sub(e[2], e[1], &e[1]);

    /*단위 벡터*/
    coef = 1.0 / g[0].x;
    vec3_scale( coef, g[0], &g[0]);
    vec3_scale( coef, e[0], &e[0]);
    coef = 1.0 / g[1].y;
    vec3_scale( coef, g[1], &g[1]);
    vec3_scale( coef, e[1], &e[1]);
    coef = 1.0 / g[2].z;
    vec3_scale( coef, g[2], &g[2]);
    vec3_scale( coef, e[2], &e[2]);


    printf("GAUSS ELIMINATION\n");
    printf("A\n");
    print_vec3(g[0]);
    print_vec3(g[1]);
    print_vec3(g[2]);

    printf("E\n");
    print_vec3(e[0]);
    print_vec3(e[1]);
    print_vec3(e[2]);

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


#endif
```