1351063

Trần Tinh Chí

Readme – Lab04

Bài này em làm thành công 2 hàm Multiply và Dot Product giữa 2 Matrices.

Hàm Inverse Matrix em làm thành công các hàm thành phần, và giờ chỉ còn bước ráp lại thôi (cũng ngắn) nhưng vì không kịp giờ nên em đưa các thầy xem code C++ lẵn code MIPS, code C++ có ghi rõ cách ráp hàm Inverse. Mong các thầy thông cảm vì bài dài quá mà sức em có hạn, nên mong các thầy chấm nương tay, đừng trừ điểm em nặng quá. Em làm cực khổ lắm .

**Inverse:**

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

using namespace std;

double Determinant(double \*\*a,int n)

{

int i,j,j1,j2;

double det = 0;

double \*\*m = NULL;

if (n < 1) { /\* Error \*/

} else if (n == 1) { /\* Shouldn't get used \*/

det = a[0][0];

} else if (n == 2) {

det = a[0][0] \* a[1][1] - a[1][0] \* a[0][1];

} else {

det = 0;

for (j1=0;j1<n;j1++) {

m = (double \*\*) malloc((n-1) \* sizeof(double \*));

for (i=0;i<n-1;i++)

m[i] = (double \*) malloc((n-1) \* sizeof(double));

for (i=1;i<n;i++) {

j2 = 0;

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

if (j == j1)

continue;

m[i-1][j2] = a[i][j];

j2++;

}

}

det += pow(-1.0,j1+2.0) \* a[0][j1] \* Determinant(m,n-1);

for (i=0;i<n-1;i++)

free(m[i]);

free(m);

}

}

return(det);

}

/\*

Find the cofactor matrix of a square matrix

\*/

void CoFactor(double \*\*a,int n,double \*\*b)

{

int i,j,ii,jj,i1,j1;

double det;

double \*\*c;

c = (double \*\*) malloc((n-1)\*sizeof(double \*));

for (i=0;i<n-1;i++)

c[i] = (double \*) malloc((n-1)\*sizeof(double));

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

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

/\* Form the adjoint a\_ij \*/

i1 = 0;

for (ii=0;ii<n;ii++) {

if (ii == i)

continue;

j1 = 0;

for (jj=0;jj<n;jj++) {

if (jj == j)

continue;

c[i1][j1] = a[ii][jj];

j1++;

}

i1++;

}

/\* Calculate the determinate \*/

det = Determinant(c,n-1);

/\* Fill in the elements of the cofactor \*/

b[i][j] = pow(-1.0,i+j+2.0) \* det;

}

}

for (i=0;i<n-1;i++)

free(c[i]);

free(c);

}

/\*

Transpose of a square matrix, do it in place

\*/

void Transpose(double \*\*a,int n)

{

int i,j;

double tmp;

for (i=1;i<n;i++) {

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

tmp = a[i][j];

a[i][j] = a[j][i];

a[j][i] = tmp;

}

}

}

int main () {

double \*\* a;

double \*\* cofactor;

int n ;

cin >> n ;

a = new double \* [n];

cofactor = new double \* [n];

for (int i = 0 ; i < n ; ++i) {

a[i] = new double [n];

cofactor[i] = new double [n];

for (int j = 0 ; j < n ; ++j) {

cin >> a[i][j];

}

}

double det = Determinant(a, n);

CoFactor(a, n, cofactor);

Transpose(cofactor, n);

for (int i = 0 ; i < n ; ++i) {

for (int j = 0 ; j < n ; ++j) {

cout << cofactor[i][j] / det << " ";

}

cout << endl;

}

return 0;

}

**Multiply Matrix:**

Matrix A : x \* y

Matrix B : m \* n

**if**(y==m)

    {

**for**(i=0;i<x;i++)

        {

**for**(j=0;j<n;j++)

            {

                c[i][j]=0;

**for**(**int** k=0;k<m;k++)

                {

                    c[i][j]=c[i][j]+a[i][k]\*b[k][j];

                }

            }

        }

        cout<<"\n-----------------------------------------------------------\n";

        cout<<"\n\nMultiplication of Matrix A and Matrix B :\n\n";

**for**(i=0;i<x;i++)

        {

**for**(j=0;j<n;j++)

            {

                cout<<"\t"<<c[i][j];

            }

            cout<<"\n\n";

        }

    }

**else**

    {

        cout<<"\n\nMultiplication is not possible";

    }