ภาคผนวก

ภาคผนวก

โปรแกรมสำหรับการทดสอบการคำนวณเชิงตัวเลขด้วยวิธีการซิงค์กาเลอร์คินบนโดเมนรูปพัด

ในการทดลองเชิงตัวเลขได้ทำการเขียนโปรแกรม Sinc_Fan เพื่อทำการทดสอบวิธีการซิงค์กาเลอร์คินบนโดเมนรูปพัดกับปัญหาสมการปัวส์ซองตามที่กล่าวไว้ในบทที่ 5 แสดงโค้ดที่เขียนเพื่อทำการทดลองดังต่อนี้

ตัวอย่างโค้ดโปรแกรม Sinc_Fan

```
#include <stdio.h>
#include <math.h>
#include "util.h"
#include "matrix.h"
#include "Legendre.h"
#include "SincSystem.h"
#include "Problem.h"
#include "Onedimension.h"
#include "Twodimension.h"
void LSinc2DSemi(double,double,double*,double*);
void LSinc2DAB(double,double,double*,double*);
main()
{
        int i,j,k,l,m,n,np,mp,N;
        int Ms,Ns,ms,Mt,Nt,mt;
        double r0,aph,h,xi,error;
        double hs,ht,Lms,Lns,Lmt,Lnt;

        r0 = 1.0; aph = 4.0*Pi/2.0;
        LSinc2DSemi(r0,aph,&Lms,&Lns);
        LSinc2DAB(r0,aph,&Lmt,&Lnt);
        print(Lmt);  print(Lnt);
        hs = 0.235;  ht = 0.095;
        for(N = 2;N <= 30;N+=2){
```

```
        Ms = N;   Ns = N;
        ms = Ns+Ms+1;
        Mt = 2*N/2;  Nt = 2*N/2;
        mt = Nt+Mt+1;
        vector x(ms),ss(ms),svi(ms),sus(ms);
        vector y(mt),tt(mt),tvi(mt),sut(mt),bd(mt);
        vector lams(ms),lamt(mt),pdass(ms),pdast(mt);
        matrix Qs(ms,ms),Qt(mt,mt),Ltt(mt,mt),uk(ms,mt),us(ms,mt);
        matrix F(ms,mt),U(ms,mt);
        hs = (Lns-Lms)/(ms-1);
        ht = (Lnt-Lmt)/(mt-1);
        for(k = -Ms;k <= Ns;k++)
                x[Ms+k+1] = spss((double)k*hs);
        for(k = -Ms;k <= Ns;k++)
                ss[Ms+k+1] = (double)k*hs;
        for(k = -Mt;k <= Nt;k++)
                y[Mt+k+1] = spst((double)k*ht,0.0,aph);
        for(k = -Mt;k <= Nt;k++)
                tt[Mt+k+1] = (double)k*ht;
        for(j = 1;j <= mt; j++)
                bd[j] = ur(r0,y[j],aph);
        for(i = 1;i <= ms;i++)
                for(j = 1;j <= mt;j++)
                        U[i][j] = ut(x[i],y[j],r0,aph);
EigenValuesProblem(x,ss,y,tt,0.0,aph,hs,ht,Qs,Qt,lams,lamt,Ltt,pdass,pdast);
uk =  CoefSincSemi2D(Ltt,Qs,Qt,lams,lamt,pdass,pdast,bd,x,y,r0,aph);
        for(i = 1;i <= ms;i++)
                for(j = 1;j <= mt;j++)
                        us[i][j] = uk[i][j]+bd[j]*swph(x[i]);

        printf("%d %e \n",ms*mt,Max(Abs(us-U)));
    }
}
```

```
void LSinc2DSemi(double r0,double aph,double *Lms,double *Lns)
{
        int i,j,imax = 200;
        double l,ll,zero;
        vector a(imax);
        for(i = 1;i <= imax;i++){
                l = 0.03*i;
                for(j = 1;j <= imax;j++){
                        ll = 0.01*j;
                        a[j] = ut(spss(l),spst(ll,0.0,aph),r0,aph);
                }
                zero = Max(a);
                if(fabs(zero) < 1.0e-16)
                 break;
        }
        *Lns = l;
         for(i = 1;i <= imax;i++){
                 l = -0.01*i;
                 for(j = 1;j <= imax;j++){
                         ll = 0.1*j;
                         a[j] = ut(spss(l),spst(ll,0.0,aph),r0,aph);
                 }
                 zero = Max(a);
                  if(fabs(zero) < 1.0e-16)
                  break;
         }
        *Lms = l;
}
void LSinc2DAB(double r0,double aph,double *Lms,double *Lns)
{
        int i,j,imax = 200;
        double l,ll,zero;
        vector a(imax);

         for(i = 1;i <= imax;i++){
```

```
                l = 0.03*i;
                for(j = 1;j <= imax;j++){
                        ll = 0.01*j;
                        a[j] = ut(spss(ll),spst(l,0.0,aph),r0,aph);
                }
                zero = Max(a);
                if(fabs(zero) < 1.0e-16)
                break;
        }

        *Lns = l;
        for(i = 1;i <= imax;i++){
                l = -0.01*i;
                for(j = 1;j <= imax;j++){
                        ll = 0.1*j;
                        a[j] = ut(spss(ll),spst(l,0.0,aph),r0,aph);
                }
                zero = Max(a);
                if(fabs(zero) < 1.0e-16)
                        break;
        }
        *Lms = l;
}

void EigenValuesProblem                         // แก้ปัญหาสมการเชิงเส้นด้วยค่าเฉพาะ
(
        vector x,vector xx,vector y,vector yy,double a,double b,double hx,
        double hy,matrix Qx,matrix Qy,vector lamx,vector lamy,matrix Lyy,
        vector pdashx,vector pdashy
)
{
        int i,j,mx = x.rows,my = y.rows;
        vector ddx(mx),ds(mx);
        matrix I2x(mx,mx),Lx(mx,mx),Lxx(mx,mx),Ax(mx,mx);
        vector ddy(my),dt(my);
        matrix I2y(my,my),Ly(my,my),Ay(my,my);
```

```
        I2x = I2(mx);
        I2y = I2(my);
        for(i = 1;i <= mx; i++)
                pdashx[i] = 1.0/spds(xx[i]);
         for(i = 1;i <= my; i++)
                pdashy[i] = 1.0/spdt(yy[i],a,b);
        for(i = 1;i <= mx; i++)
                ddx[i] = sqrt(pdashx[i]);
         for(i = 1;i <= my; i++)
                ddy[i] = sqrt(pdashy[i]);
        for(i = 1;i <= mx; i++)
                ds[i] = digs(xx[i]);
        for(i = 1;i <= my; i++)
                dt[i] = digt(yy[i]);
        Lx = (-1.0/(hx*hx)*I2x+DiagonalMatrix(ds));
        Ly = (-1.0/(hy*hy)*I2y+DiagonalMatrix(dt));
        Lyy = Ly*DiagonalMatrix(ddy);
        Ax = DiagonalMatrix(pdashx)*Lx*DiagonalMatrix(pdashx);
        Ay = DiagonalMatrix(pdashy)*Ly*DiagonalMatrix(pdashy);


        JEigenSystem(Ax,lamx,Qx);
        JEigenSystem(Ay,lamy,Qy);
        eigsrt(lamx,Qx);
        eigsrt(lamy,Qy);
}

matrix CoefSincSemi2D                    // แก้ปัญหาสมการเชิงเส้น
(
        matrix Lyy,matrix Qx,matrix Qy,vector lamx,vector lamy,vector pdashx,
        vector pdashy,vector bd,vector x,vector y,double r0,double aph)
{
        int i,j,mx = x.rows,my = y.rows;
        matrix F(mx,my),FN(mx,my),G(mx,my),H(mx,my)
        matrix W(mx,my),V(mx,my),U(mx,my);
        matrix P(mx,my),Q(mx,my);
```

```
        vector sx(mx),sy(my);
        vector prootx(mx),prooty(my),dfy(my);

        for(i = 1;i <= mx; i++)
                prootx[i] = sqrt(1.0/pdashx[i]);
        for(i = 1;i <= my; i++)
                prooty[i] = sqrt(1.0/pdashy[i]);
        for(i = 1;i <= my; i++)
                dfy[i] = pow(pdashy[i],1.5);
        for(i = 1;i <= mx; i++)
                for(j = 1;j <= my; j++)
                        P[i][j] = swph(x[i])*bd[j];
        for(i = 1;i <= mx; i++)
                for(j = 1;j <= my; j++)
                        Q[i][j] = swphd(x[i])*bd[j];

        for(i = 1;i <= mx;i++)
                for(j = 1;j <= my;j++)
                        F[i][j] = ft(x[i],y[j],r0,aph);

        FN= (F-P*Transpose(Lyy)*DiagonalMatrix(dfy)-Q);

        G = DiagonalMatrix(prootx)*FN*DiagonalMatrix(prooty);
        H = Transpose(Qx)*G*Qy;

        for(i = 1;i <= mx; i++)
                for(j = 1;j <= my; j++)
                        W[i][j] = H[i][j]/(lamx[i]+lamy[j]);

        for(i = 1;i <= mx; i++)
                sx[i] = 1.0/prootx[i];
        for(j = 1;j <= my; j++)
                sy[j] = 1.0/prooty[j];

        V = Qx*W*Transpose(Qy);
        U = DiagonalMatrix(sx)*V*DiagonalMatrix(sy);
        return U;
}
```

```
double ft(double eta,double th,double r0,double aph)
{
        return exp(-2.0*(eta-log(r0)))*fr(exp(-eta+log(r0)),th,aph);
}

double ur(double r,double th,double aph)
{
        return r*(1.0-r)*sin(Pi*th/aph);
}

double ut(double eta,double th,double r0,double aph)
{
        return ur(exp(-eta+log(r0)),th,aph);
}

double lpss(double s,double L)
{
        return sinh(L*acos(s)/Pi)*sinh(L*acos(s)/Pi);
}

double spst(double x,double a,double b)
{
        return  ((b*pow(E,0.5*Pi*sinh(x)) +a*pow(E,-0.5*Pi*sinh(x)))/(pow(E,-
                0.5*Pi*sinh(x))+pow(E,0.5*Pi*sinh(x))));
}

double spss(double x)
{
        return exp(x-exp(-x));
}

double spds(double x)
{
        return ((1+exp(x))/exp(exp(-x)));
}
```

```
double spdt(double x,double a,double b)
{
        return (-(a-b)*pow(E,Pi*sinh(x))*Pi*cosh(x)/pow(1.0+pow(E,Pi*sinh(x)),2.0));
}

double spsroot(double x)
{
        return sqrt(spds(x));
}

double sptroot(double x,double a,double b)
{
        return sqrt(spdt(x,a,b));
}

double digs(double x)
{
        return (7.0 + exp(-2.0*x)+4.0/exp(x)+2.0*exp(x)
                +exp(2.0*x))/(4.0*(1.0+exp(x))*(1.0+exp(x)));
}

double digt(double x)
{
         return 0.25*(-2.0+ Pi*Pi*cosh(x)*cosh(x)+3.0*tanh(x)*tanh(x));
}

double swph(double x)
{
         return (exp(-x)*(1.0+x));
}
double swphd(double x)
{
        return (-exp(-x)*(x-1.0));
}
```

//โค้ดคลาส vector

```
class vector
{
public:
        int rows;
        double *x;

        vector(int r);
        vector(const vector&);
        ~vector() {}
        int size(){ return rows;}
        double& operator[](const int i) { return x[i]; }
        double  operator[](const int i) const   { return x[i]; }
        void print(void);
        double norm_2(void);

        vector& operator=(const vector& v);
        vector& operator=(const double& d);
        vector& operator+=(const vector&);
        vector& operator-=(const vector&);

        friend vector operator+(const vector&, const vector&);          // vector+vector
        friend vector operator-(const vector&, const vector&);          // vector-vector
        friend double operator*(const vector&,const vector&);           // vector*vector
        friend vector operator*(const vector& v, const double d);       // vector*scalar
        friend vector operator*(const double d,const vector& v)         // scalar*vector
                { return v * d; }
        friend int operator==(const vector&,const vector&);             // equality
};
vector::vector(int r)
{
        if(r < 0){
                cout << "error\n";
                exit(0);
        }
        rows = r;
```

```
        x = new double[r+1];
}

vector::vector(const vector& other_vector)
{
        rows = other_vector.rows;
        x = other_vector.x;
}


vector& vector::operator=(const vector &v)
{
        if(this == &v)
                return *this;
        for(int i = 1; i <= rows; i++)
                x[i] = v.x[i];
                return *this;
}

vector& vector::operator=(const double &d)
{
        for(int i = 1; i <= rows; i++)
                x[i] = d;
        return *this;
}
vector& vector::operator=(const double &d)
{
        for(int i = 1; i <= rows; i++)
                x[i] = d;
        return *this;
}

vector& vector::operator+=(const vector &a)                    // v += a
{
        if ( a.rows != rows)
        {
```

```
                cout << "vector a += vector b different sizes, a = " << rows
                << ", b = " << a.rows;
                exit(0);
        }
        int sz = rows;
        for (int i = 1; i <= sz; i++)
                ( *this )[i] += a[i];
        return *this;
}

vector& vector::operator-=(const vector &a)          // v -= a
{
        if ( a.rows != rows)
        {
                cout << "vector a += vector b different sizes, a = " << rows
                    << ", b = " << a.rows;
                exit(0);
        }
        int sz = rows;
        for (int i = 1; i <= sz; i++)
                ( *this )[i] -= a[i];
        return *this;
}

vector operator+(const vector& a, const vector &b)          //  v = a + b
{
        if ( a.rows != b.rows)
        {
                cout << "vector a + vector b different sizes, a = " << a.rows
                << ", b = " << b.rows;
                exit(0);
        }
        int sz = a.rows;
        vector sum(sz);
        for (int i = 1; i <= sz; i++)
                sum.x[i] = a.x[i] + b.x[i];
        return sum;
}
```

```
vector operator-(const vector& a, const vector &b)          //  v = a- b
{
        if ( a.rows != b.rows)
        {
                cout << "vector a + vector b different sizes, a = " << a.rows
                     << ", b = " << b.rows;
                exit(0);
        }
        int sz = a.rows;
        vector sum(sz);
        for (int i = 1; i <= sz; i++)
                sum.x[i] = a.x[i] - b.x[i];
        return sum;
}

double operator*(const vector &a,const vector &b)          //  v = a * b
{
        if ( a.rows != b.rows)
        {
                cout << "vector a * vector b incommensurate, a = " << a.rows
                << ", b = " << b.rows;
                exit(0);
        }
        int i, sz = a.rows;
        double prod, zero = 0.0;
        for (i = 1, prod = zero; i <= sz; i++)
                prod += a.x[i] * b.x[i];
        return prod;
}

vector operator*(const vector& v, const double d)          // v = a * d
{
        int i, sz = v.rows;
        vector  b(sz) ;
```

```
        for (i = 1; i <= sz; i++)
                b.x[i] = v.x[i]*d;

        return b;

}

int operator==(const vector &a,const vector &b)      //        int i = (a == b)
{
        if ( a.rows != b.rows )
        {
                cout << "vector a and vector b different sizes, a = " << a.rows
                        << ", b = " << b.rows;
                exit(0);
        }

        int i, sz = a.rows;
        int l = 1;
        for (i = 1; i <= sz; i++)
                l = l && ( a[i] == b[i] );
        return l;
}
```

//โค้ด คลาส matrix

```
class matrix
{
public:
        double **m;
        int rows,cols;
        matrix(const int,const int);                    // constructor
        matrix(const matrix&);                          // copy
        ~matrix(){}                                     // destructor
        matrix& operator=(const matrix&);               // substution
        double* operator[](const int i){return m[i];}
```

```
        double* operator[](const int i) const {return m[i];}
        int getrows(void){ return rows;}
        int getcols(void){ return cols;}
        void resize(int r,int c){ rows = r;cols = c;}          // resize
        void print(void);                                      // print elements
        void diag(const double fv);                            // set diagonals
        vector Rownumber(const int i);
        vector Colnumber(const int j);

        friend matrix operator+(const matrix&,const matrix&);      // Matrix+Matrix
        friend matrix operator-(const matrix&,const matrix&);      // Matrix-Matrix
        friend matrix operator*(const matrix&,const matrix&);      // Matrix*Matrix
        friend matrix operator*(const double&,const matrix&);      // double*Matrix
        friend vector operator*(const matrix&,vector&);            // Matrix*vector
        friend int operator==(const matrix&,const matrix&);         // equality
};

matrix& matrix::operator=(const matrix &a)                 // กำหนดค่าเมทริกซ์ให้เมทริกซ์
{
        int i, j;
        if ( this == &a )
                return *this;
        if ( a.rows != rows || a.cols != cols )
        {
                if ( rows != a.rows )
                        cout << "rows = " << rows << ", a.rows = " << a.rows;
                if ( cols != a.cols)
                        cout << " cols = " << cols << ", a.cols = " << a.cols;
                exit(0);
                }
                for (i = 1; i <= rows; i++)
                for (j = 1; j <= cols; j++)
                        m[i][j] = a.m[i][j];

                return *this;
}
```

```
void matrix::print(void)
{
        int i,j;
        for(i = 1; i <= rows; i++){
                for(j = 1; j <= cols; j++)
                        printf("%6.3lf ", m[i][j]);
                        printf("%18.16lf ", m[i][j]);
                        printf("\n");
        }
        printf("\n");
}

void matrix::diag(const double a)                    // สร้างเมทริกซ์ทะแยงมุมด้วยสเกลาร์
{
        int i, iend;
        iend = rows;
        if ( iend > cols )
                iend = cols;
        for (i = 1; i <= iend; i++)
                m[i][i] = a;
}

matrix operator+(const matrix &a,const matrix &b)          // เมทริกซ์บวกเมทริกซ์
{
        if ( a.rows != b.rows || a.cols != b.cols )
        {
                if ( a.rows != b.rows )
                        cout << " a.row = " << a.rows << ", b.row = " << b.rows;
                if ( a.cols != b.cols)
                        cout << " a.col = " << a.cols << ", b.col = " << b.cols;
        exit(0);
        }

        int i, j, row = a.rows, col = a.cols;
        matrix sum(row,col);
        for (i = 1; i <= row; i++)
```

```
                for (j = 1; j <= col; j++)
                        sum.m[i][j] = a.m[i][j] + b.m[i][j];
        return sum;
}
matrix operator-(const matrix &a,const matrix &b)          // เมทริกซ์ลบเมทริกซ์
{
        int i, j;
        if ( a.rows != b.rows || a.cols != b.cols )
        {
                if ( a.rows != b.rows )
                        cout << " a.row = " << a.rows << ", b.row = " << b.rows;
                if ( a.cols != b.cols)
                        cout << " a.col = " << a.cols << ", b.col = " << b.cols;
                        exit(0);
        }
        int row = a.rows, col = a.cols;
        matrix diff(row,col);
        for (i = 1; i <= row; i++)
                for (j = 1; j <= col; j++)
                        diff.m[i][j] = a.m[i][j] - b.m[i][j];

        return diff;
}
matrix operator*(const matrix &a,const matrix &b)          //  เมทริกซ์คูณเมทริกซ์
{
        if ( a.cols != b.rows )
        {
                cout << "matrix a * matrix b incommensurate, a.cols = " << a.cols
                        << " , b.rows = " << b.rows;
                exit(0);
        }
        int i, j, k, row = a.rows, col = b.cols, size = a.cols;
        matrix prod(row,col);
        double zero = 0.0;
        for (i = 1; i <= row; i++)
                for (j = 1; j <= col; j++)
```

```
                    for (k = 1, prod.m[i][j] = zero; k <= size; k++)
                        prod.m[i][j] += a.m[i][k] * b.m[k][j];
    return prod;
}

matrix  operator*(const double  &d,const matrix &a)        // จำนวนจริงคูณเมทริกซ์
{
    int i, j, row = a.rows, col = a.cols;
    matrix b(row,col);
    for (i = 1; i <= row; i++)
        for (j = 1; j <= col; j++)
            b.m[i][j] = d * a.m[i][j];
    return b;
}
int operator==(const matrix &a,const matrix &b)     //เปรียบเทียบค่าเมทริกซ์
{
    if ( a.rows != b.rows || a.cols != b.cols )
    {
        if ( a.rows != b.rows )
        cout << "matrix a and matrix b different sizes  a.row = " << a.rows
            << ", b.row = " << b.rows;
        if ( a.cols != b.cols)
        cout << "matrix a and matrix b different sizes, a.col = " << a.cols
            << ", b.col = " << b.cols;
        exit(0);
    }
    int i, j, row = a.rows, col = a.cols;
    int l = 1;
    for (i = 1; i <= row; i++)
        for (j = 1; j <= col; j++)
            l = l && ( a.m[i][j] == b.m[i][j] );
    return l;
}
matrix DiagonalMatrix(vector a)
{
```

```
	int i,j,n = a.rows;
	matrix A(n,n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++)
			A[i][j] = ((i == j) ? a[i]:0.0);
	return A;
}

matrix DiagonalMatrix(double a,int n)
{
	int i,j;
	matrix A(n,n);
	for (i = 1; i <= n; i++)
		for (j = 1; j <= n; j++)
			A[i][j] = ((i == j) ? a:0.0);
	 return A;
}
matrix Abs(matrix M)
{
	int i,j,r = M.rows,c = M.cols;
	matrix A(r,c);
	for(i = 1; i <= r; i++)
		for(j = 1; j <= c; j++)
			A[i][j] = fabs(M[i][j]);

	return A;
}
void JEigenSystem(matrix a,vector d,matrix v)
{
	int j,iq,ip,i,n=a.rows;
	double tresh,theta,tau,t,sm,s,h,g,c;
	vector b(n),z(n);
	for (ip=1;ip<=n;ip++) {
		for (iq=1;iq<=n;iq++)v[ip][iq]=0.0;
			v[ip][ip]=1.0;
	}
```

```
for (ip=1;ip<=n;ip++) {
        b[ip]=d[ip]=a[ip][ip];
        z[ip]=0.0;
}
for (i=1;i<=100;i++) {
        sm=0.0;
        for (ip=1;ip<=n-1;ip++) {
                for (iq=ip+1;iq<=n;iq++)
                        sm += fabs(a[ip][iq]);
        }
        if (sm == 0.0) {
                z = ZeroVector(n);
                b = ZeroVector(n);
                return;
        }
        if (i < 4)
                tresh=0.2*sm/(n*n);
        else
                tresh=0.0;
        for (ip=1;ip<=n-1;ip++) {
                for (iq=ip+1;iq<=n;iq++) {
                        g=100.0*fabs(a[ip][iq]);
                        if (i > 4 && (float)(fabs(d[ip])+g) == (float)fabs(d[ip])
                        && (float)(fabs(d[iq])+g) == (float)fabs(d[iq]))
                                a[ip][iq]=0.0;
                        else if (fabs(a[ip][iq]) > tresh) {
                                h=d[iq]-d[ip];
                                if ((float)(fabs(h)+g) == (float)fabs(h))
                                        t=(a[ip][iq])/h;
                                else {
                                        theta=0.5*h/(a[ip][iq]);
                                        t=1.0/(fabs(theta)+sqrt(1.0+theta*theta));
                                if (theta < 0.0) t = -t;
                                }
                                c=1.0/sqrt(1+t*t);
                                s=t*c;
```

```
                                tau=s/(1.0+c);
                                h=t*a[ip][iq];
                                z[ip] -= h;
                                z[iq] += h;
                                d[ip] -= h;
                                d[iq] += h;
                                a[ip][iq]=0.0;

                                for (j=1;j<=ip-1;j++) {
                                        ROTATE(a,j,ip,j,iq)
                                }
                                 for (j=ip+1;j<=iq-1;j++) {
                                        ROTATE(a,ip,j,j,iq)
                                }
                                for (j=iq+1;j<=n;j++) {
                                        ROTATE(a,ip,j,iq,j)
                                }
                                For (j=1;j<=n;j++) {
                                        ROTATE(v,j,ip,j,iq)
                                }
                        }
                    }
                }
                for (ip=1;ip<=n;ip++) {
                        b[ip] += z[ip];
                        d[ip]=b[ip];
                        z[ip]=0.0;
                }
        }
        printf("Too many iterations in routine jacobi\n");
}
void eigsrt(vector d,matrix v)
{
        int k,j,i,n = d.rows;
        double p;
        for (i=1;i<n;i++) {
                p=d[k=i];
```

```
                for (j=i+1;j<=n;j++)
                        if (d[j] >= p) p=d[k=j];
                if (k != i) {
                        d[k]=d[i];
                        d[i]=p;
                        for (j=1;j<=n;j++) {
                                p=v[j][i];
                                v[j][i]=v[j][k];
                                v[j][k]=p;
                        }
                }
        }
}

vector ZeroVector(int n)        //สร้างเวกเตอร์ศูนย์
{
        int i,j;
        vector Zero(n);
        for(i = 1;i <= n; i++)
                Zero[i] = 0.0;
        return  Zero;
}

ROTATE(a,i,j,k,l)
{
                g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);a[k][l]=h+s*(g-h*tau);
}

matrix Transpose(matrix M)
{
        int i, j, r = M.cols, c = M.rows;
        matrix A(r,c);
        for (i = 1; i <= r; i++)
                for (j = 1; j <= c; j++)
                        A[i][j] = M[j][i];
        return A;
}
```

ประวัติผู้เขียน

ชื่อ-สกุล นายปราการ ศรีงาม

วัน เดือน ปี เกิด 2 มิถุนายน 2519

ประวัติการศึกษา สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย โรงเรียนกำแพงเพชรพิทยาคม จังหวัดกำแพงเพชร ปีการศึกษา 2536

สำเร็จการศึกษาระดับปริญญาวิทยาศาสตร์บัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2541

ประสบการณ์ ผู้ช่วยนักวิจัย โครงการ การประมาณผลผลิตอ้อยด้วยแบบจำลองคอมพิวเตอร์ ระยะที่ 2 ศูนย์วิจัยเพื่อเพิ่มผลผลิตทางเกษตร มหาวิทยาลัยเชียงใหม่ ปี พ.ศ.2540

ผู้ช่วยนักวิจัย โครงการการประมาณผลผลิตอ้อยด้วยแบบจำลองคอมพิวเตอร์ ระยะที่ 3 ศูนย์วิจัยเพื่อเพิ่มผลผลิตทางเกษตร มหาวิทยาลัยเชียงใหม่ ปี พ.ศ.2542

ผู้ช่วยนักวิจัย โครงการการพัฒนาและบำรุงรักษาโฮมเพจ สำนักงานกองทุนสนับสนุนงานวิจัย สำนักงานภาค มหาวิทยาลัยเชียงใหม่ ปี พ.ศ. 2542

ผู้ช่วยนักวิจัย การพัฒนาระบบเว็ปไซท์ของ สกว. สำนักงานกองทุนสนับสนุนงานวิจัย ปี พ.ศ. 2545

ผู้ช่วยนักวิจัย โครงการระบบสนับสนุนการตัดสินใจผลิตพืชระดับท้องถิ่น: ท้องทุ่งไทย ระยะที่ 2-3 ศูนย์วิจัยเพื่อเพิ่มผลผลิตทางเกษตร มหาวิทยาลัยเชียงใหม่ ปี พ.ศ. 2545-2547