NATIONAL INSTITUE OF TECHNOLOGY KARANATAKA, SURATHKAL

ASSIGNMENT 2

APPLIED COMPUTATIONAL METHODS IN MECHANICAL SCIENCES

RAJAT A CHANDAVAR – 16ME156

13-Aug-19

ASSIGNMENT ON DOOLITTLE, CROWT, CHOLESKI METHODS AND THOMAS ALGORITHM

Question 1 Answer:

Information regarding the code:

Inorder to solve the given matrix it has to be first converted into lower and upper triangular matrices. Declaration of whole n x n array for these leads to wastage of computational memory. To prevent this, pointers are declared and assigned memory in the code. However, the pointers have to be mapped to their particular matrix position which is done using the functions idxl(int,int) and idxu(int,int) for lower and upper triangular matrices respectively.

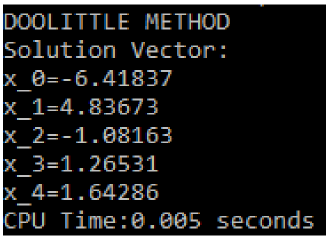
For example, in the lower triangular matrix $\begin{bmatrix} l_{00} & 0 & 0 \\ l_{10} & l_{12} & 0 \\ l_{21} & l_{22} & l_{23} \end{bmatrix}$, pointer $l[0]$ has to be mapped to l_{00}, $l[1]$ to l_{10}, $l[2]$ to l_{12}, $l[3]$ to l_{21} and so on..

Program (C++)

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
#include<math.h>
using namespace std;
int index,n=5,i,j,k,choice=2,d;
float *l,*u,z[5],x[5],sum,e;
int idxl(int i, int j)
{
    d=0;
    if(choice==1)
        d=i;
    index=(i*(i+1))/2+j-d;
    return index;
}
int idxu(int i, int j)
{
    d=0;
    if(choice==2)
        d=i+1;
    index=(i*(2*n+1-i))/2+(j-i)-d;
    return index;
}
void findz(float *l,float *z,int *b)
{
    e=1;
    for(i=0;i<n;++i)
    {
        sum=0;
        for(j=0;j<i;++j)
            sum+=l[idxl(i,j)]*z[j];
        if(choice==2)
            e=l[idxl(i,i)];
        z[i]=(b[i]-sum)/e;
    }
}
void findx(float *u,float *x,float *z)
{
    e=1;
    for(i=n-1;i>=0;--i)
    {
        sum=0;
        for(j=i+1;j<n;++j)
            sum+=u[idxu(i,j)]*x[j];
        if(choice!=2)
            e=u[idxu(i,i)];
        x[i]=(z[i]-sum)/e;
    }
}
main()
{
    clock_t start=clock();
    int b[n]={-2,4,3,-5,1}, flag=0;
    int a[n][n]={{2,1,1,3,2},{1,2,2,1,1},{1,2,9,1,5},{3,1,1,7,1},{2,1,5,1,8}};
    if(choice==1)//DOOLITTLE
    {
        cout<<"DOOLITTLE METHOD\n";
        l=(float*)calloc(n*(n-1)/2,sizeof(float));
        u=(float*)calloc(n*(n+1)/2,sizeof(float));
        for(i=0;i<n;++i)
        {
            for(k=0;k<i;++k)
            {
                sum=0;
                for(j=0;j<k;++j)
                    sum+=l[idxl(i,j)]*u[idxu(j,k)];
                l[idxl(i,k)]=(a[i][k]-sum)/u[idxu(k,k)];
            }
            for(j=i;j<n;++j)
            {
                sum=0;
                for(k=0;k<i;++k)
                    sum+=l[idxl(i,k)]*u[idxu(k,j)];
                u[idxu(i,j)]=a[i][j]-sum;
            }
        }
        findz(l,z,b);
        findx(u,x,z);
    }
    else if(choice==2)//CROUT
    {
        cout<<"CROUT METHOD\n";
        l=(float*)calloc(n*(n+1)/2,sizeof(float));
        u=(float*)calloc(n*(n-1)/2,sizeof(float));
        for(i=0;i<n;++i)
        {
            for(k=0;k<=i;++k)
            {
                sum=0;
                for(j=0;j<k;++j)
                    sum+=l[idxl(i,j)]*u[idxu(j,k)];
                l[idxl(i,k)]=a[i][k]-sum;
            }
            for(j=i+1;j<n;++j)
            {
                sum=0;
                for(k=0;k<i;++k)
                    sum+=l[idxl(i,k)]*u[idxu(k,j)];
                u[idxu(i,j)]=(a[i][j]-sum)/l[idxl(i,i)];
            }
        }
        findz(l,z,b);
        findx(u,x,z);
    }
    else//CHOLESKY
    {
        cout<<"CHOLESKY METHOD\n";
        u=(float*)calloc(n*(n+1)/2,sizeof(float));
        for(i=0;i<n;++i)
        {
            if(a[i][i]<0)
            {
                flag=1;
                break;
            }
            for(j=0;j<n;++j)
            {
                if(a[i][j]!=a[j][i])
                {
                    flag=1;
                    break;
                }
            }
            if(flag)
                break;
        }
        if(flag)
            cout<<"Matrix is not symmetric or not positive definite. Hence Cholesky method cannot be used.";
        else
        {
            for(i=0;i<n;++i)
            {
                sum=0;
                for(k=0;k<i;++k)
                    sum+=pow(u[idxu(k,i)],2);
                if(a[i][i]<sum)
                {
                    cout<<"Hello"<<i;
                    flag=1;
                    break;
                }
                u[idxu(i,i)]=sqrt(a[i][i]-sum);
                for(j=i+1;j<n;++j)
                {
                    sum=0;
                    for(k=0;k<i;++k)
                        sum+=u[idxu(k,i)]*u[idxu(k,j)];
                    u[idxu(i,j)]=(a[i][j]-sum)/u[idxu(i,i)];
                }
            }
            for(i=0;i<n;i++)
            {
                sum=0;
                for(j=0;j<i;j++)
                    sum+=u[idxu(j,i)]*z[j];
                z[i]=(b[i]-sum)/u[idxu(i,i)];
            }
            findx(u,x,z);
        }
    }
    if (!flag)
    {
        cout<<"Solution Vector:";
        for(i=0;i<n;i++)
            cout<<"\nx_"<<i<<"="<<x[i];
    }
    delete(l);delete(u);
    clock_t stop=clock();
    double timespent = (double)(stop-start)/(double)CLOCKS_PER_SEC;
    cout<<"\nCPU Time:"<<timespent<<" seconds";
}
```

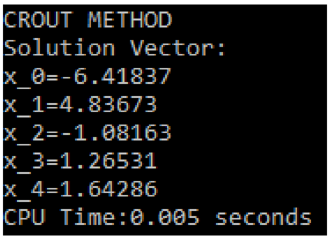
Output:

Choice 1: Doolittle



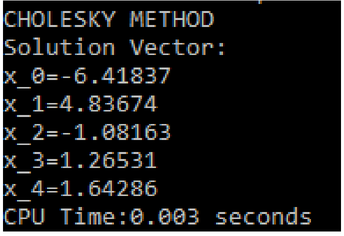
```
DOOLITTLE METHOD
Solution Vector:
x_0=-6.41837
x_1=4.83673
x_2=-1.08163
x_3=1.26531
x_4=1.64286
CPU Time:0.005 seconds
```

Choice 2: Crowt



```
CROUT METHOD
Solution Vector:
x_0=-6.41837
x_1=4.83673
x_2=-1.08163
x_3=1.26531
x_4=1.64286
CPU Time:0.005 seconds
```

Choice 3: Cholesky



```
CHOLESKY METHOD
Solution Vector:
x_0=-6.41837
x_1=4.83674
x_2=-1.08163
x_3=1.26531
x_4=1.64286
CPU Time:0.003 seconds
```

Question 2 Answer:

Discretized equations

$$k_1 T_1 - T_2 = k_2 + T_0$$

$$-T_{i-1} + k_1 T_i - T_{i+1} = k_2 \text{ , i=2 and 3}$$

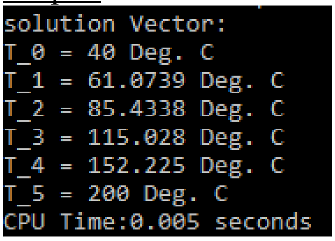
$$-T_3 + k_1 T_4 = k_2 + T_5$$

Where, $k_1 = 2 + h'\Delta x^2, k_2 = h'\Delta x^2 T_a$ and $T_0 = 40°C, T_5 = 200°C, T_a = 20°C$

Code (C++)

```
#include<iostream>
#include<time.h>
using namespace std;
main()
{
    clock_t start=clock();
    int l=10,i,j,dx=2,n=l/dx+1,Ta=20;
    float T[n],h=0.02,k1=2+h*dx*dx,k2=h*dx*dx*Ta,d[n-2],a[n-3],b[n-2],c[n-3];
    T[0]=40;
    T[n-1]=200;
    for(i=0;i<n-3;i++)
    {
        a[i]=c[i]=-1;
        b[i]=k1;
        if(!i)
            d[i]=k2+T[0];
        else
            d[i]=k2;
    }
    d[n-3]=k2+T[n-1];
    b[n-3]=k1;
    for(i=0;i<n-3;++i)
    {
        a[i]=a[i]/b[i];
        b[i+1]=b[i+1]-a[i]*c[i];
        d[i+1]=d[i+1]-a[i]*d[i];
    }
    T[n-2]=d[n-3]/b[n-3];
    for(i=n-3;i>0;--i)
        T[i]=(d[i-1]-c[i-1]*T[i+1])/b[i-1];
    cout<<"solution Vector:";
    for(i=0;i<n;i++)
        cout<<"\nT_"<<i<<" = "<<T[i]<<" Deg. C";
    clock_t stop=clock();
    double timespent = (double)(stop-start)/(double)CLOCKS_PER_SEC;
    cout<<"\nCPU Time:"<<timespent<<" seconds";
}
```

Output:



```
solution Vector:
T_0 = 40 Deg. C
T_1 = 61.0739 Deg. C
T_2 = 85.4338 Deg. C
T_3 = 115.028 Deg. C
T_4 = 152.225 Deg. C
T_5 = 200 Deg. C
CPU Time:0.005 seconds
```