

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, 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 $n \times n$ 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++)

```
1  #include<iostream>
2  #include<stdlib.h>
3  #include<time.h>
4  #include<math.h>
5  using namespace std;
6  int index,n=5,i,j,k,choice=2,d;
7  float *l,*u,z[5],x[5],sum,e;
8  int idxl(int i, int j)
9  {
10     d=0;
11     if(choice==1)
12         d=i;
13     index=(i*(i+1))/2+j-d;
14     return index;
15 }
16 int idxu(int i, int j)
17 {
18     d=0;
19     if(choice==2)
20         d=i+1;
21     index=(i*(2*n+1-i))/2+(j-i)-d;
22     return index;
23 }
24 void findz(float *l,float *z,int *b)
25 {
26     e=1;
27     for(i=0;i<n;++i)
28     {
29         sum=0;
30         for(j=0;j<i;++j)
31             sum+=l[idxl(i,j)]*z[j];
32         if(choice==2)
33             e=l[idxl(i,i)];
34         z[i]=(b[i]-sum)/e;
35     }
36 }
37 void findx(float *u,float *x,float *z)
38 {
39     e=1;
40     for(i=n-1;i>=0;--i)
41     {
42         sum=0;
43         for(j=i+1;j<n;++j)
44             sum+=u[idxu(i,j)]*x[j];
45         if(choice!=2)
46             e=u[idxu(i,i)];
47         x[i]=(z[i]-sum)/e;
```

```

48     }
49 }
50 main()
51 {
52     clock_t start=clock();
53     int b[n]={-2,4,3,-5,1}, flag=0;
54     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} };
55     if(choice==1)//DOOLITTLE
56     {
57         cout<<"DOOLITTLE METHOD\n";
58         l=(float*)calloc(n*(n-1)/2,sizeof(float));
59         u=(float*)calloc(n*(n+1)/2,sizeof(float));
60         for(i=0;i<n;++i)
61         {
62             for(k=0;k<i;++k)
63             {
64                 sum=0;
65                 for(j=0;j<k;++j)
66                     sum+=l[idxl(i,j)]*u[idxu(j,k)];
67                 l[idxl(i,k)]=(a[i][k]-sum)/u[idxu(k,k)];
68             }
69             for(j=i;j<n;++j)
70             {
71                 sum=0;
72                 for(k=0;k<i;++k)
73                     sum+=l[idxl(i,k)]*u[idxu(k,j)];
74                 u[idxu(i,j)]=a[i][j]-sum;
75             }
76         }
77         findz(l,z,b);
78         findx(u,x,z);
79     }
80     else if(choice==2)//CROUT
81     {
82         cout<<"CROUT METHOD\n";
83         l=(float*)calloc(n*(n+1)/2,sizeof(float));
84         u=(float*)calloc(n*(n-1)/2,sizeof(float));
85         for(i=0;i<n;++i)
86         {
87             for(k=0;k<=i;++k)
88             {
89                 sum=0;
90                 for(j=0;j<k;++j)
91                     sum+=l[idxl(i,j)]*u[idxu(j,k)];
92                 l[idxl(i,k)]=a[i][k]-sum;
93             }
94             for(j=i+1;j<n;++j)
95             {
96                 sum=0;
97                 for(k=0;k<i;++k)
98                     sum+=l[idxl(i,k)]*u[idxu(k,j)];
99                 u[idxu(i,j)]=a[i][j]-sum/l[idxl(i,i)];
100             }
101         }
102         findz(l,z,b);
103         findx(u,x,z);
104     }
105     else//CHOLESKY
106     {
107         cout<<"CHOLESKY METHOD\n";
108         u=(float*)calloc(n*(n+1)/2,sizeof(float));
109         for(i=0;i<n;++i)
110         {
111             if(a[i][i]<0)
112             {
113                 flag=1;
114                 break;
115             }

```

```

116         for(j=0;j<n;++j)
117         {
118             if(a[i][j]!=a[j][i])
119             {
120                 flag=1;
121                 break;
122             }
123         }
124         if(flag)
125             break;
126     }
127     if(flag)
128         cout<<"Matrix is not symmetric or not positive definite. Hence
Cholesky method cannot be used.";
129     else
130     {
131         for(i=0;i<n;++i)
132         {
133             sum=0;
134             for(k=0;k<i;++k)
135                 sum+=pow(u[idxu(k,i)],2);
136             if(a[i][i]<sum)
137             {
138                 cout<<"Hello"<<i;
139                 flag=1;
140                 break;
141             }
142             u[idxu(i,i)]=sqrt(a[i][i]-sum);
143             for(j=i+1;j<n;++j)
144             {
145                 sum=0;
146                 for(k=0;k<i;++k)
147                     sum+=u[idxu(k,i)]*u[idxu(k,j)];
148                 u[idxu(i,j)]=(a[i][j]-sum)/u[idxu(i,i)];
149             }
150         }
151         for(i=0;i<n;i++)
152         {
153             sum=0;
154             for(j=0;j<i;j++)
155                 sum+=u[idxu(j,i)]*z[j];
156             z[i]=(b[i]-sum)/u[idxu(i,i)];
157         }
158         findx(u,x,z);
159     }
160 }
161 if (!flag)
162 {
163     cout<<"Solution Vector:";
164     for(i=0;i<n;i++)
165         cout<<"\nx_"<<i<<"="<<x[i];
166 }
167 delete(l);delete(u);
168 clock_t stop=clock();
169 double timespent = (double)(stop-start)/(double)CLOCKS_PER_SEC;
170 cout<<"\nCPU Time:"<<timespent<<" seconds";
171 }

```

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

$$\begin{aligned}k_1 T_1 - T_2 &= k_2 + T_0 \\ -T_{i-1} + k_1 T_i - T_{i+1} &= k_2, i=2 \text{ and } 3 \\ -T_3 + k_1 T_4 &= k_2 + T_5\end{aligned}$$

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

Code (C++)

```
1  #include<iostream>
2  #include<time.h>
3  using namespace std;
4  main()
5  {
6      clock_t start=clock();
7      int l=10,i,j,dx=2,n=l/dx+1,Ta=20;
8      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];
9      T[0]=40;
10     T[n-1]=200;
11     for(i=0;i<n-3;i++)
12     {
13         a[i]=c[i]=-1;
14         b[i]=k1;
15         if(!i)
16             d[i]=k2+T[0];
17         else
18             d[i]=k2;
19     }
20     d[n-3]=k2+T[n-1];
21     b[n-3]=k1;
22     for(i=0;i<n-3;++i)
23     {
24         a[i]=a[i]/b[i];
25         b[i+1]=b[i+1]-a[i]*c[i];
26         d[i+1]=d[i+1]-a[i]*d[i];
27     }
28     T[n-2]=d[n-3]/b[n-3];
29     for(i=n-3;i>0;--i)
30         T[i]=(d[i-1]-c[i-1]*T[i+1])/b[i-1];
31     cout<<"solution Vector:";
32     for(i=0;i<n;i++)
33         cout<<"\nT_"<<i<<" = "<<T[i]<<" Deg. C";
34     clock_t stop=clock();
35     double timespent = (double)(stop-start)/(double)CLOCKS_PER_SEC;
36     cout<<"\nCPU Time:"<<timespent<<" seconds";
37 }
```

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
```