Jaime Frade
Computational Finance: Dr. Kopriva
Homework #4

0. **Executive Summary**

1. **Statement of Problem**

2. Description of the Mathematics

3. Description of the Algorithm

4. **Results**

5. Conclusion

APPENDIX: 1 Main file

```
#include <iostream>
using namespace std;
#include <iomanip>
using namespace std;
#include <math.h>
#include <vector>
using namespace std;
#include <algorithm>
using namespace std;

#include "tridiag2.h"

int solveTridiagonal(double *, double *, double *, double *, double *, int);
void coeffcubepoly(double *y, double *d, double *z, double *t, int n, int x);
int main()
{
tridiag2 t1;
const int n=8;
double t[8] = {0.5, 1, 2, 4, 5,10, 15, 20};
double y[8] = {0.0552, 0.060, 0.0682, 0.0801, 0.0843,0.0931, 0.0912, 0.0857};
double array[n] ={0};
double array2[n] ={0};
double array3[n] ={0};
double array4[8];




for (int i=0; i<=n-1; i++)
{
array[i] = t[i+1] - t[i];
array2[i] = 2*(array[i+1]-array[i]);
array3[i] = 6*(y[i+1]-y[i])/(array[i]);
array4[i] = array3[i] - array3[i-1];
}
array4[0] = 0;
array4[7] = 0;
double *aPtr= array;
double *dPtr = array2;
double *cPtr = array;
double *RPtr = array3;
double *xPtr = array4;
double *yPtr = y;
double *tPtr = t;

t1.solveTridiagonal(aPtr, dPtr, cPtr, RPtr,xPtr, n);

int x1=.5;
int x2=1;
```

```
int x3 =2;
int x4 =4;
int x5= 5;
int x6=10;
int x7=15;
int x8=20;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x1, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x2, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x3, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x4, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x5, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x6, n);
cout << endl;cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x7, n);
cout << endl;
coeffcubepoly(yPtr, dPtr, xPtr, tPtr, x8, n);


return 0;

}


void coeffcubepoly(double *y1, double *d1, double *z1, double *t1, int x, int n)
{
double A[8] ={0};
double B[8] ={0};
double C[8] ={0};
double S[8] ={0};

for(int i=0; i<=n-1; i++)
{
A[i]=(z1[i+1]-z1[i])/(6*d1[i]);
B[i] = z1[i]/2;
C[i] = (-d1[i]/6)*z1[i+1] + (-d1[i]/3)*z1[i] + (1/d1[i])*(y1[i+1]-y1[i]);

S[i] = y1[i] + (x-t1[i])*(C[i]+(x-t1[i])*(B[i])+(x-t1[i])*A[i]);

cout << S[i] << endl;
}
 }
```

APPENDIX: 2 tridiag2.cpp

```
//Function that solves tridiagonal matrix
//using Thomas algortihm
//member function definitions for class tridiag2

//provides formula for spline in each SUBinterval


#include <iostream>
using namespace std;
#include <iomanip>
using namespace std;
#include <math.h>

//include definition of class tridiag2 from tridiag2.h
#include "tridiag2.h"


int tridiag2::solveTridiagonal(double *a, double *d, double *c, double *R, double *x, int n)
{

//do forward elmination, gauss elmin for submatrix
for(int i = 1; i < n; i++)
{
if (d[i-1] == 0.0)
return 1;
        d[i] -= a[i]*c[i-1]/d[i-1];
        R[i] -= a[i]*R[i-1]/d[i-1];
}
//do back substitution

if (d[n-1] == 0.0)
return 2;

    for (x[n-1] = R[n-1]/d[n-1], i = n-2; i >= 0; i--)
{
        if(d[i] == 0.0)
return 2;
        x[i] = (R[i] - c[i]*x[i+1])/d[i];
    }
return 0;
} //end function solveTridiagonal
```

APPENDIX: 3 tridiag2.h

```
//Declaration of class tridiag2
//member function is defined in tridiag2.cpp

class tridiag2 {

public:
//Tridiag();
int solveTridiagonal(double *, double *, double *, double *, double *, int);
private:
};  //end class tridiag2
```