//本程序是求解二阶线性常微分方程两点Dirichlet边值问题的紧差分方法：

//y’’+q(x)y=f(x), a<x<b, y(a)=alpha, y(b)=beta

#include "stdafx.h"

#include<math.h>

#include<stdio.h>

#include<stdlib.h>

int m;

void main()

{

double \*matrix\_a\_array, \*matrix\_b\_array, \*matrix\_c\_array, \*x, \*z, \*rhs, \*ans, \*y;

double a, b, h, Pi, alpha, beta, c;

int i, j;

double f(double x); //原方程右端项函数f(x)

double \*farray(double \*x); //f(x)在各节点的函数值组成的数组

double q(double x); //原方程中的函数q(x)

double \*chase\_algorithm(double \*a, double \*b, double \*c, int n, double \*d);

//追赶法子程序，a,b,c分别为系数矩阵的下次、主、上次对角线元素组，n为方程组阶数，d为矩阵右端项向量

double exact(double x);

m = 16;

Pi = 3.14159265359;

a = 0.0; //边界左端点

b = Pi / 2.0; //边界右端点

h = (b - a) / m;

alpha = 0.0;

beta = 0.0;

x = (double \*)malloc(sizeof(double)\*(m + 1));

for (i = 0; i <= m; i++)

{

x[i] = a + i\*h;

}

z = farray(x);

c = h\*h / 12.0;

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

for (i = 1; i<m; i++) //矩阵的右端项，含m-1个元素的数组rhs

{

rhs[i - 1] = (z[i - 1] + 10 \* z[i] + z[i + 1])\*c;

}

rhs[0] = rhs[0]-alpha\*(1 + c\*q(x[0])); //考虑边界条件

rhs[m - 2] = rhs[m - 2]-beta\*(1 + c\*q(x[m]));

free(z);

matrix\_a\_array = (double \*)malloc(sizeof(double)\*(m - 1)); //矩阵的下次对角线

matrix\_b\_array = (double \*)malloc(sizeof(double)\*(m - 1)); //矩阵的主对角线

matrix\_c\_array = (double \*)malloc(sizeof(double)\*(m - 1)); //矩阵的上次对角线

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

{

matrix\_a\_array[i] = 1.0 + c\*q(x[i]);

matrix\_b\_array[i] = 10 \* c\*q(x[i + 1]) - 2;

matrix\_c\_array[i] = 1.0 + c\*q(x[i + 2]);

}

ans = chase\_algorithm(matrix\_a\_array, matrix\_b\_array, matrix\_c\_array, m - 1, rhs);

free(matrix\_a\_array); free(matrix\_b\_array); free(rhs);

y = (double \*)malloc(sizeof(double)\*(m + 1)); //y为数值解

y[0] = alpha;

for (i = 1; i<m; i++)

y[i] = ans[i - 1];

free(ans);

y[m] = beta;

j = m / 4;

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

printf("x=%.2f, ynumerical=%f,exact=%f, err=%.4e\n", x[i], y[i], exact(x[i]), fabs(exact(x[i]) - y[i]));

}

double f(double x)

{

return exp(x)\*(2\*cos(x)-sin(x));

}

double \* farray(double \*x)

{

int i;

double \*z;

z = (double \*)malloc(sizeof(double)\*(m + 1));

for (i = 0; i <= m; i++)

z[i] = f(x[i]);

return z;

}

double q(double x)

{

return -1.0;

}

double \* chase\_algorithm(double \*a, double \*b, double \*c, int n, double \*d)

{

double \* ans, \*g, \*w, p;

int i;

ans = (double \*)malloc(sizeof(double)\*n);

g = (double \*)malloc(sizeof(double)\*n);

w = (double \*)malloc(sizeof(double)\*n);

g[0] = d[0] / b[0];

w[0] = c[0] / b[0];

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

{

p = b[i] - a[i] \* w[i - 1];

g[i] = (d[i] - a[i] \* g[i - 1]) / p;

w[i] = c[i] / p;

}

ans[n - 1] = g[n - 1];

i = n - 2;

do

{

ans[i] = g[i] - w[i] \* ans[i + 1];

i = i - 1;

} while (i >= 0);

free(g);

free(w);

return ans;

}

double exact(double x)

{

return sin(x);

}