C 语言编程习题

第二章

习题 2-2

5. 用二分法编程求 $6x^4-40x^2+9=0$ 的所有实根。

```
#include <stdio.h>
#include <math.h>

#define N 10000

double A,B,C;

double f(double x)
{
    return (A*x*x*x*x+B*x*x+C);
}

void BM(double a,double b,double eps1,double eps2)
{
    int k;
    double x,xe;
    double valuea = f(a);
    double valueb = f(b);
    if (valuea > 0 && valueb > 0 || valuea <0 && valueb < 0) return;

    printf("Finding root in the range: [%.3lf, %.3lf]\n", a, b);
    for(k=1;k<=N;k++) {
        x=(a+b)/2;
        xe=(b-a)/2;
        if(fabs(xe)<eps2 || fabs(f(x))<eps1) {
            printf("The x value is:%g\n",x);
            printf("f(x)=%g\n\n",f(x));
            return;
        }
        if(f(a)*f(x)<0) b=x;
        else a=x;
    }
    printf("No convergence!\n");
}
```

```
int main()
{
    double a,b,eps1,eps2,step,start;

    printf("Please input A,B,C:\n");
    scanf("%lf %lf %lf",&A,&B,&C);

    printf("Please input a,b, step, eps1,eps2:\n");
    scanf("%lf %lf %lf %lf %lf",&a,&b,&step,&eps1,&eps2);

    for (start=a; (start+step) <= b; start += step) {
        double left = start;
        double right = start + step;
        BM(left, right, eps1, eps2);
    }

    return 0;
}
```

运行：

Please input A,B,C:
6 -40 9

Please input a,b, step, eps1,eps2:
-10 10 1 1e-5 1e-5

Finding root in the range: [-3.000, -2.000]
The x value is:-2.53643
f(x)=-0.00124902

Finding root in the range: [-1.000, 0.000]
The x value is:-0.482857
f(x)=0.00012967

Finding root in the range: [0.000, 1.000]
The x value is:0.482857
f(x)=0.00012967

Finding root in the range: [2.000, 3.000]
The x value is:2.53643
f(x)=-0.00124902

有时若把判别语句

```
if(fabs(xe)<eps2 || fabs(f(x))<eps1)
```
改为
```
if(fabs(xe)<eps2 && fabs(f(x))<eps1)
```
会提高精度，对同一题运行结果：

Finding root in the range: [-3.000, -2.000]
The x value is:-2.53644
f(x)=-4.26496e-007

Finding root in the range: [-1.000, 0.000]
The x value is:-0.482861
f(x)=-7.3797e-006

Finding root in the range: [0.000, 1.000]
The x value is:0.482861
f(x)=-7.3797e-006

Finding root in the range: [2.000, 3.000]
The x value is:2.53644
f(x)=-4.26496e-007

习题 2-3

5. 请用埃特金方法编程求出 x=tgx 在 4.5（弧度）附近的根。

```
#include <stdio.h>
#include <math.h>

#define N 100
#define PI 3.1415926

void SM(double x0,double eps)
{
    int k;
    double x;
    double x1,x2;

    for(k=1;k<=N;k++) {
        x1=sin(x0)/cos(x0);
        x2=sin(x1)/cos(x1);

        x=x0-(x1-x0)*(x1-x0)/(x2-2*x1+x0);
        if(fabs(x-x0)<eps) {
```

```
            printf("The x value is:%g\n",x);
            return;
        }
        x0=x;
    }
    printf("No convegence!\n");
}

int main()
{
    double eps,x0;
    printf("Please input eps,x0:\n");
    scanf("%lf %lf",&eps,&x0);
    SM(x0,eps);
    return 0;
}
```

运行:

Please input eps,x0:
1e-5 4.5
The x value is:4.49341

习题 2-4

11. 请编出用牛顿法求复根的程序，并求出

$$P(z)=z^4-3z^3+20z^2+44z+54=0$$

接近于 $z_0=2.5+4.5i$ 的零点。

```
#include <cstdio>
#include <cmath>

#define MAX_TIMES 1000
typedef struct {
      double real, image;
} COMPLEX;

COMPLEX Aa[5]={{54,0},{44,0},{20,0},{-3,0},{1,0}};
COMPLEX Bb[4]={{44,0},{40,0},{-9,0},{4,0}};
COMPLEX zero = {0, 0};
```

```
double eps1=1e-6;
double eps2=1e-6;

COMPLEX multi(COMPLEX a,COMPLEX b)
{
	COMPLEX result;
	result.real = a.real * b.real - a.image * b.image;
	result.image = a.image * b.real + a.real * b.image;
	return result;
}
COMPLEX Div(COMPLEX a,COMPLEX b){
	COMPLEX z3;
	double s;
	s=(b.real*b.real)+(b.image*b.image);

	z3.real=b.real;
	z3.image=-b.image;
	z3=multi(a,z3);
	z3.real=z3.real/s;
	z3.image=z3.image/s;

	return z3;
}
COMPLEX add(COMPLEX a,COMPLEX b)
{
	COMPLEX result;
	result.real = a.real + b.real;
	result.image = a.image + b.image;
	return result;
}
COMPLEX subtract(COMPLEX a, COMPLEX b)
{
	COMPLEX result;
	result.real = a.real - b.real;
	result.image = a.image - b.image;
	return result;
}
COMPLEX times(COMPLEX z,int n){
	int i;
	COMPLEX result={1, 0};
	for (i=0;i<n;i++) result=multi(result,z);
	return result;
}
```

```
double distance(COMPLEX a, COMPLEX b) {
	return sqrt((a.real - b.real) * (a.real - b.real) + (a.image - b.image) * (a.image - b.image));
}

double complex_abs(COMPLEX a) {
	return sqrt(a.real * a.real + a.image * a.image);
}

COMPLEX f(COMPLEX x)
{
	int i;
	COMPLEX result=zero;
	for (i=0;i<5;i++){
		result=add(result,multi(Aa[i],times(x,i)));
	}
	return result;
}

COMPLEX ff(COMPLEX x) {
	int i;
	COMPLEX result=zero;
	for (i=0;i<4;i++){
		result=add(result,multi(Bb[i],times(x,i)));
	}
	return result;
}

int main(){
	COMPLEX z0,z1,result;
	double x,y;
	int k;

	printf("please input x,y\n");
	scanf("%lf %lf",&x,&y);

	z0.real=x; z0.image=y;

	for(k=0; k<MAX_TIMES; k++) {
		z1 = subtract(z0, Div(f(z0), ff(z0)));
		result = f(z1);
		if (distance(z0,z1)<eps1 || complex_abs(result)<eps2){
			printf("The root is: z=(%.3f + %.3fi), f(z)=(%e + %ei)\n", z1.real,z1.image,
result.real, result.image);
			return 0;
```

```
            }
            z0 = z1;
        }
        printf("No convergence!\n");
        return 0;
}
```

运行:
please input x,y
2.5 4.5
The root is:
z=(2.471 + 4.641i), f(z)=(-1.122705e-007 + -1.910245e-007i)

习题 2-6

2. 请编程用劈因子法求高次方程 $x^4+x^3+5x^2+4x+4=0$ 的所有复根。

```
#include<cstdio>
#include<cmath>

int main()
{
        float b[20],c[20];
        float delta;
        float eps;
        int print=0;
        float fru0,fru1,s0,s1,frv0,frv1;
        float deltau,deltav;
        float u,v;
        float a[20];
        int n,j;
        float r0,r1;
        float r;

        int i, k;

        printf("Please input max exponent:\n");
        scanf("%d",&n);
        printf("Please input epslion:\n");
        scanf("%f",&eps);
        printf("Please input the coefficient:\n");
        for(i=0;i<=n;i++) scanf("%f,",&a[i]);
```

```
for(j=0;j<=n;j++) printf("+%.3fX^%d",a[j],n-j);
printf("\n");


for(k=1;k<=2;k++) {
	u=a[n-1]/a[n-2];
	v=a[n]/a[n-2];
	if(k==2) {
		u=0;v=4;
	}

	while(1) {
		b[0]=a[0];
		b[1]=a[1]-u*b[0];
		for(j=2;j<=n;j++) {
			b[j]=a[j]-u*b[j-1]-v*b[j-2];
		}
		r0=b[n-1];
		r1=b[n]+u*b[n-1];

		c[0]=b[0];
		c[1]=b[1]-u*b[0];
		for(j=2;j<=n-2;j++)
			c[j]=b[j]-u*c[j-1]-v*c[j-2];
		s0=c[n-3];
		s1=c[n-2]+u*c[n-3];
		fru0=u*s0-s1;
		fru1=v*s0;
		frv0=-s0;
		frv1=-s1;

		deltau=-(frv1*r0-frv0*r1)/(fru0*frv1-fru1*frv0);
		deltav=-(fru1*r0-fru0*r1)/(frv0*fru1-frv1*fru0);
		u=u+deltau;
		v+=deltav;
		delta=u*u-4*v;
		if((fabs(deltau)<eps)&&(fabs(deltav)<eps))
		{
			print=1;
			printf("found roots:");
			r=-u/2;
			i=(int) sqrt(fabs(delta))/2;
			if(delta<0) {
```

```
                    printf("%.3f+%.3fi\t",r,fabs((double)i));
                    printf("%.3f-%.3fi\n",r,fabs((double)i));
                }
                else {
                    printf("%.3f\t\t",r+i);
                    printf("%.3f\n",r-i);
                }
                if(n==1) printf("Found root :%.3f\n",a[1]/a[0]);
                if(k==2) return 0;
            }
            if(k==1&&print) break;
        }   /* end while */
    }   /* end for */
    return 0;
}
```

运行:

Please input max exponent:
4
Please input epslion:
1e-5
Please input the coefficient:
1 1 5 4 4
+1.000X^4+1.000X^3+5.000X^2+4.000X^1+4.000X^0
found roots:-0.500+0.000i -0.500-0.000i
found roots:0.000+2.000i 0.000-2.000i

习题 3-1

5. 请编程用列全主元高斯—约当消去法求矩阵 A, B 的逆矩阵。

$$A=\begin{bmatrix}-3 & 8 & 5\\ 2 & -7 & 4\\ 1 & 9 & -6\end{bmatrix} \qquad B=\begin{bmatrix}2 & 1 & -3 & -1\\ 3 & 1 & 0 & 7\\ -1 & 2 & 4 & -2\\ 1 & 0 & -1 & 5\end{bmatrix}$$

```
#include <cstdio>
#include <cmath>
#define MAX   10
int ROW;
```

```
static float A[MAX+1][2*MAX+1];

void putout()
{
	int i,j;
	printf("\nThe reverse matrix is:\n");
	for(i=1;i<=ROW;i++) {
		for(j=ROW+1;j<=2*ROW;j++) {
			printf("%.3f ",A[i][j]);
		}
		printf("\n");
	}
}

int get()
{
	float max[MAX+1];
	int count_r[MAX+1];
	float tmp[2*MAX+1];
	float pass1,pass2;
	int i,j,k;
	for(i=1;i<=ROW;i++) {
		max[i]=A[i][i];
		count_r[i]=i;
		for(k=i;k<=ROW;k++) {
			if(max[i]<A[k][i]) {
				max[i]=A[k][i];
				count_r[i]=k;
			}
		} /********** find the max one**********/

		if(max[i]==0) return -1;
		if(count_r[i]!=i) {
			for(k=1;k<=2*ROW;k++)
			{
				tmp[k]=A[count_r[i]][k];
				A[count_r[i]][k]=A[i][k];
				A[i][k]=tmp[k];
			}
		} /*********change the rows*************/

		pass1=A[i][i];
		for(k=1;k<=2*ROW;k++) {
			A[i][k]/=pass1;
```

```
            }

            for(j=1;j<=ROW;j++) {
                if(j==i) continue;
                else {
                    pass2=A[j][i];
                    for(k=1;k<=2*ROW;k++) {
                        A[j][k]=A[j][k]-pass2 *A[i][k];
                    }
                }
            } /************ get simple**************/
        }
        putout();
        return 0;
}

int main()
{
        int i,j;
        float tmp=-1;
        float p;
        printf("please input the number of ROW:");
        scanf("%d",&ROW);
        printf("\n");

        if(ROW>=MAX) {
            printf("the    number of row is too big!\n");
            return 0;
        }

        printf("Please input the    matrix A:\n");
        for(i=1;i<=ROW;i++) {
            for(j=1;j<=ROW;j++) {
                scanf("%f,",&p);
                A[i][j]=p;
                if(tmp<fabs(A[i][j])) tmp=fabs(A[i][j]);
            }
        }
        if(tmp==0) {
            printf("No inverse!");
            return 0;
        }

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

```
            A[i][i+ROW]=1;
        }
        return get();
}
```

运行：

please input the number of ROW:
3
Please input the matrix A:
-3 8 5
2 -7 4
1 9 -6
The reverse matrix is:
0.026 0.396 0.285
0.068 0.055 0.094
0.106 0.149 0.021

please input the number of ROW:
4
Please input the matrix A:
2 1 -3 -1
3 1 0 7
-1 2 4 -2
1 0 -1 5
The reverse matrix is:
-0.047 0.588 -0.271 -0.941
0.388 -0.353 0.482 0.765
-0.224 0.294 -0.035 -0.471
-0.035 -0.059 0.047 0.294

习题 3-2

4. 编程用追赶法解下列三对角线方程组。

$$\begin{bmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 100 \\ 200 \\ 200 \\ 200 \\ 100 \end{bmatrix}$$

```
#include <stdio.h>
```

```
#include <math.h>
#define    ROW       5
static float A[ROW+1]={0,0,0,0,0,0};
static float B[ROW+1]={0,0,0,0,0,0};
static float C[ROW+1]={0,0,0,0,0,0};
static float X[ROW+1]={0,0,0,0,0,0};
int main()
{
      int i;
      printf("Please input the A:\n");
      for(i=2;i<=ROW;i++) scanf("%f,",&A[i]);

      printf("Please input the B:\n");
      for(i=1;i<=ROW;i++) scanf("%f,",&B[i]);

      printf("Please input the C:\n");
      for(i=1;i<=ROW-1;i++) scanf("%f,",&C[i]);

      printf("Please input the F:\n");
      for(i=1;i<=ROW;i++) scanf("%f,",&X[i]);

      C[1]=C[1]/B[1];
      for(i=2;i<=ROW-1;i++)
            C[i]=C[i]/(B[i]-A[i]*C[i-1]);
      X[1]=X[1]/B[1];
      for (i=2; i<=ROW; i++)
            X[i]=(X[i]-A[i]*X[i-1])/(B[i]-A[i]*C[i-1]);
      for(i=ROW-1;i>=1;i--)
            X[i]=X[i]-C[i]*X[i+1];

      printf ("The value is : \n");
      for(i=1;i<=ROW;i++) printf ("x%d=%.3f\n",i,X[i]);

      return 0;
}
```

运行：

Please input the A:
-1 -1 -1 -1
Please input the B:
4 4 4 4 4
Please input the C:
-1 -1 -1 -1
Please input the F:

100 200 200 200 100
The value is :
x1=46.154
x2=84.615
x3=92.308
x4=84.615
x5=46.154

习题 4-3

1. 用高斯—赛德尔方法编程解下列线性方程组，要求当 $\| x^{(k+1)} - x^{(k)} \| < 10^{-5}$ 时迭代终止。

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 6 \\ -2 \\ 6 \end{bmatrix}$$

```
#include<cstdio>
#include<cmath>
float x[6];
float y[6];
float eps;
float a[6][6]={
    {4,-1,0,-1,0,0},
    {-1,4,-1,0,-1,0},
    {0,-1,4,0,0,-1},
    {-1,0,0,4,-1,0},
    {0,-1,0,-1,4,-1},
    {0,0,-1,0,-1,4}
};
float b[6]={0,5,0,6,-2,6};
int gs()
{
    int i,k,j;
    float s;
    for(i=0;i<6;i++) y[i]=x[i]=0;
```

```
    for(k=0;k<20;k++) {
        for(i=0;i<6;i++) {
            s=0;
            for(j=0;j<6;j++) {
                if(j!=i) s=s+a[i][j]*y[j];
            }
            y[i]=(b[i]-s)/a[i][i];
        }
        for(i=0;(i<6)&&(abs(y[i]-x[i])<eps);i++) ;
        if(i>=5) break;
        for(j=0;j<6;j++) x[j]=y[j];

        if(k>20) {
            printf("No convergence./n");
            return -1;
        }
    }
    return 1;
}

int main()
{
    int tag,i,m;
    printf("\nthe max circles=");
    scanf("%d",&m);
    printf("eps=");
    scanf("%lf",&eps);

    tag=gs();
    if(tag>0) {
        for(i=0;i<=5;i++) {
            x[i]=y[i];
            printf("x(%d)=%.3e\n",i+1,x[i]);
        }
    }
    return 0;
}
```

运行：

nthe max circles=

10

eps=

1e-5

x(1)=1.000e+000

x(2)=2.000e+000
x(3)=1.000e+000
x(4)=2.000e+000
x(5)=1.000e+000
x(6)=2.000e+000

习题 4-4

11. 上机用松弛法解方程组:

$$\begin{cases} 4x_1 - x_2 = 1 \\ -x_1 + 4x_2 - x_3 = 4 \\ -x_2 + 4x_3 = -3 \end{cases}$$

取初值 $x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$，分别取$\omega = 1, \omega = 1.1, \omega = 1.03,$

要求当 $\left\| x^* - x^{(k)} \right\|_\infty < 5 \times 10^{-4}$时迭代终止 。

```
#include <cmath>
#include <cstdio>
int agsdl(double a[][10], double b[],int n,
              double x[],double eps,double w,int m)
{
      int i,j,pcount;
      double p,t,s,q;

      for(i=0;i<10;i++) x[i]=0;

      p=eps+1.0;
      for (pcount=0;p>=eps&&pcount<m;pcount++) {
            p=0.0;
            for (i=0; i<n; i++) {
                  t=x[i]; s=0.0;
                  for (j=0; j<n; j++)
                        if (j!=i) s=s+a[i][j]*x[j];
                  x[i]=(b[i]-s)/a[i][i];
                  q=w*fabs(x[i]-t);
                  if (q>p) p=q;
            }
```

```
    }
    return pcount;
}

int main()
{
    int i,j,n,m,count=0;
    double eps,w[3];
    static double a[10][10];
    static double x[10],b[10];

    printf("\nthe max circles=");
    scanf("%d",&m);
    printf("n=");
    scanf("%d",&n);
    printf("\nmatrix A:\n");

    for(i=0;i<n;i++)
        for(j=0;j<n;j++) scanf("%lf",&a[i][j]);

    printf("\nvector b:\n");
    for(i=0;i<n;i++) scanf("%lf",&b[i]);
    printf("eps=");
    scanf("%lf",&eps);

    for(i=0;i<3;i++) {
        printf("w(%d)=",i);
        scanf("%lf",&w[i]);
    }

    for(j=0;j<3;j++) {
        printf("w(%d)=%.3f\n",j,w[j]);
        if(agsdl(a,b,n,x,eps,w[j],m)<m) {
            for(i=0;i<n;i++) printf("x(%d)=%.3e    ",i,x[i]);
            printf("\n");
        }
        else
        {
            printf("fail\n");
        }
    }
    return 0;
}
```

运行：

```
the max circles=
10
n=
3
matrix A:
4    -1    0
-1    4    -1
0    -1    4
vector b:
1    4    -3
eps=
1e-5
w(0)=
1.03
w(1)=
1
w(2)=
1.1
w(0)=1.030
x(0)=5.000e-001    x(1)=1.000e+000    x(2)=-5.000e-001
w(1)=1.000
x(0)=5.000e-001    x(1)=1.000e+000    x(2)=-5.000e-001
w(2)=1.100
x(0)=5.000e-001    x(1)=1.000e+000    x(2)=-5.000e-001
```

第 5 章

习题 5-1

2. 编程求下列矛盾方程组的解:

$$\begin{cases} x_1 + x_2 + x_3 = 4 \\ 4x_1 + 2x_2 + x_3 = 10 \\ 9x_1 + 3x_2 + x_3 = 18 \\ 16x_1 + 4x_2 + x_3 = 26 \end{cases}$$

```
#include<fstream.h>
#include<stdio.h>
#include<stdlib.h>
```

```
double eps=1.0e-5;//error range
double **a; //save matrix A
double **ab;//save augmented matrix AB
double **ta;//save matrix ATA
double *b;    //save vectors B
double *tb; //save ATB
double **ca;//save matrix A
double *cb; //save vectors B
double *x;    //save result vectors X
int *fr;//row flag
int *fc;//column flag
int m,cm;//row num of matrix A
int n;//column num of matrix A
int r;

void init();
int G_J();
int d_0(int);
void change();
void g_a_m();
void show();
void show1();

void main()
{     int t;

      init();
      t=1;
      while(1)
      {     r=G_J();
            if (m==r) break;
            if (d_0(r)==0) break;
            change();
            t=0;
      }
      if (n==r)
      {     if (t)
            {     cout<<"该线性方程组是非矛盾线性方程组，有唯一解："<<endl;
                  show();
            }
            else
            {     cout<<"该线性方程组是矛盾线性方程组，有唯一最小二乘解："<<endl;
                  show();
            }
```

```
        }
        else
        {    if (t)
             {    cout<<"该线性方程组是非矛盾线性方程组，有普遍解："<<endl;
                  show1();
             }
             else
             {    cout<<"该线性方程组是矛盾线性方程组，有普遍最小二乘解："<<endl;
                  show1();
             }
        }
}

void init()
{   //open the data file---ab.txt
        ifstream fin("ab.txt");
        if(!fin)
        {    cout<<"Can't open file ab.txt to read!"<<endl;
             exit(-1);
        }
        //read the row and column information of matrix A
        fin>>m>>n;
        cm=m;
        //require room of matrix A
        a=(double **)malloc((m+1)*sizeof(double *));
        int i,j;
        for(i=1;i<=m;i++)
             a[i]=(double *)malloc((n+1)*sizeof(double));
        //read matrix A's data from ab.txt
        for(i=1;i<=m;i++)
             for(j=1;j<=n;j++)
                  fin>>a[i][j];
        //init matrix B
        b=(double*)malloc((m+1)*sizeof(double));
        for(i=1;i<=m;i++)
             fin>>b[i];
        //close the data file
        fin.close();
        //backup matrix A and B
        ca=a; cb=b;
        return;
}

//generate augmented matrix
```

```
void g_a_m()
{     int i,j;
      //require room
      ab=(double **)malloc((m+1)*sizeof(double*));
      for(i=1;i<=m;i++)
            ab[i]=(double *)malloc((n+2)*sizeof(double));
      //load data
      for(i=1;i<=m;i++)
      {     for(j=1;j<=n;j++)
                  ab[i][j]=a[i][j];
            ab[i][n+1]=b[i];
      }
      //init the flag array fr(flag row) and fc(flag column)
      fr=(int*)malloc((m+1)*sizeof(int));
      fc=(int*)malloc((n+1)*sizeof(int));
      for(i=1;i<=m;i++) fr[i]=i;
      for(j=1;j<=n;j++) fc[j]=j;
      return;
}

//Gauss-Jordon elimination method
int G_J()
{     int k,found,min;
      int i,j,pi,pj,ft;
      double t1,t2;

      //generate augmented matrix
      g_a_m();
      k=1; min=(m<n)?m:n;
      //look for main element
      while(k<=min)
      {     t1=found=pi=pj=0;
            for(i=k;i<=m;i++)
                  for(j=k;j<=n;j++)
                  {     //t2=abs(ab[fr[i]][fc[j]]);
                        t2=ab[fr[i]][fc[j]];
                        t2=(t2>0)?t2:(-t2);
                        if((t2>t1)&&(t2>eps))
                        {     found=1; t1=t2;
                              pi=i; pj=j;
                        }
                  }//end of j
            //end of i
            if(!found)
```

```
                return k-1;
            //found a main element
            //fix the flag array---exchange the row and column
            ft=fr[pi]; fr[pi]=fr[k]; fr[k]=ft;
            ft=fc[pj]; fc[pj]=fc[k]; fc[k]=ft;
            //row standardize
            for(j=1;j<=n+1;j++)
                ab[fr[k]][j]/=t1;
            //elimination
            for(i=1;i<=m;i++)
            {   if(i==k) continue;
                t2=ab[fr[i]][fc[k]];
                for(j=1;j<=n+1;j++)
                    ab[fr[i]][j]-=ab[fr[k]][j]*t2;
            }
            //look for next main element
            k++;
        }//end of while
        return k-1;
}

//a=aTa, b=aTb
void change()
{
        //require room for aTa
        ta=(double**)malloc((n+1)*sizeof(double*));
        int i;
        for(i=1;i<=n;i++)
            ta[i]=(double*)malloc((n+1)*sizeof(double));
        //computing aTa
        int j,k;
        double sum;
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
            {   sum=0;
                for(k=1;k<=m;k++)
                    sum+=a[k][i]*a[k][j];
                ta[i][j]=sum;
            }
        //require roo for aTb
        tb=(double*)malloc((n+1)*sizeof(double));
        //computing aTb
        for(i=1;i<=n;i++)
        {   sum=0;
```

```
            for(k=1;k<=m;k++)
                sum+=a[k][i]*b[k];
            tb[i]=sum;
        }
        //free a,b
        //a=aTa, b=aTb
        a=ta; b=tb;
        m=n;
        return;
}

//if from d(r+1) to d(m) are all zero, return 0
//otherwise,return 1
int d_0(int r)
{
        int j;
        int t0;
        float t1;

        t0=0;
        for(j=r+1;j<=m;j++)
        {   t1=ab[fr[j]][n+1];
            if(t1>eps||t1<-eps)
                t0=1;
        }
        return t0;
}


//show the augmented matrix AB
void show()
{   int i,j;

        //output and store the result X[0..n-1]
        x=(double*)malloc(n*sizeof(double));
        for(j=1;j<=n;j++)
            for(i=1;i<=m;i++)
                if(ab[i][j]==1)
                {   x[j-1]=ab[i][n+1];
                    cout<<"X["<<j-1<<"]="<<x[j-1]<<endl;
                }
        //output the residual vectors
        cout<<"该方程组的剩余向量是："<<endl;
        cout<<"r=(";
```

```
    double sum;
    for(i=1;i<=cm;i++)
    {   sum=0;
        for(j=1;j<=n;j++)
            sum+=ca[i][j]*x[j-1];
        cout<<cb[i]-sum;
        if(i!=cm) cout<<",";
    }
    cout<<")T"<<endl;
    return;
}

void show1()
{   int i,j,k,found;

    for(i=1;i<=r;i++)
    {   cout<<"X["<<fc[i]<<"]="<<ab[fr[i]][n+1];
        for(j=1;j<=n-r;j++)
            cout<<"-X["<<j<<"]*("<<ab[fr[i]][fc[r+j]]<<")";
        cout<<endl;
    }
    for(i=r+1;i<=n;i++)
        cout<<"X["<<fc[i]<<"]=X["<<fc[i]<<"]"<<endl;
    cout<<"是否需要特解?(y/n)";
    char ch;
    cin>>ch;
    if((ch=='n')||(ch=='N')) return;
    cout<<"请输入参数："<<endl;
    //input variables
    x=(double*)malloc((n+1)*sizeof(double));
    for(i=r+1;i<=n;i++)
    {   cout<<"X["<<fc[i-r]<<"]=";
        cin>>x[i];
    }
    //computing special result
    for(i=1;i<=r;i++)
    {   x[fc[i]]=ab[fr[i]][n+1];
        for(j=1;j<=n-r;j++)
            x[fc[i]]-=x[r+j]*ab[fr[i]][fc[r+j]];
    }
    //output special result
    cout<<"该方程组的特解是："<<endl;
    for(i=1;i<=n;i++)
        cout<<"X["<<i<<"]="<<x[i]<<endl;
```

```
    //output the residual vectors
    cout<<"该方程组的剩余向量是："<<endl;
    cout<<"r=(";
    double sum;
    for(i=1;i<=cm;i++)
    {   sum=0;
        for(j=1;j<=n;j++)
            sum+=ca[i][j]*x[j];
        cout<<cb[i]-sum;
        if(i!=cm) cout<<",";
    }
    cout<<")T"<<endl;
    return;
}
```

运行：
ab.txt：
4 3
1 1 1
4 2 1
9 3 1
16 4 1
4 10 18 26

该线性方程组是矛盾线性方程组，有唯一最小二乘解：
X[0]=0.5
X[1]=4.9
X[2]=-1.5
该方程组的剩余向量是：
r=(0.1,-0.3,0.3,-0.1)^T

第六章

习题 6-1

9. 已知正弦函数表

x_k	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
Sin x_k	0.4794	0.6442	0.7833	0.8912	0.9636	0.9975	0.9917	0.9463

编程用 Lagrange 插值多项式计算 x_0=0.6,0.8,1.0 处的函数值 sin(0.6), sin(0.8), sin(1.0) 的近似值$f(0.6),f(0.8),f(1.0)$。

```
#include <cstdio>
#include <cmath>
#include <cstdlib>

#define N 8

float x[N]={0.5f, 0.7f, 0.9f, 1.1f, 1.3f, 1.5f, 1.7f, 1.9f };
float f[N]={0.4794f,0.6442f,0.7833f,0.8912f,0.9636f,0.9975f,0.9917f,0.9463f};

float lagrange(float x0)
{
	float s;
	float result = 0.0;
	int j,k;

	for(k=0;k<N;k++) {
		s=1.0;
		for(j=0;j<N;j++) {
			if(j!=k) s *= (x0-x[j])/(x[k]-x[j]);
		}
		result += f[k]*s;
	}
	return result;
}

int main(void)
{
	float x0,f0;
	printf("Input the value of x0: ");
	scanf("%f",&x0);
	float result = lagrange(x0);
	printf("x0=%.3f    f(x0)=%.3f    sin(x0)=%.3f\n", x0, result, sin(x0));
	return 0;
}
```

运行:

Input the value of x0:
x0=0.600   f(x0)=0.565   sin(x0)=0.565
Input the value of x0:
x0=0.800   f(x0)=0.717   sin(x0)=0.717
Input the value of x0:
x0=1.000   f(x0)=0.841   sin(x0)=0.841

习题 6-5

7.已知某沿海货轮 475 水线的型值表, n=15

i	x_i	y_i
1	1162.5	99.5
2	2950	220.2355
3	5900	589.83954
4	7675.8515	950
5	8850	1260.2342
6	10881.255	1900
7	11880	2207.0194
8	13802.899	2850
9	14750	3124.1306
10	17467.481	3800
11	17700	3853.2175
12	20650	4390.3286
13	23600	4756.1508
14	26550	4976.2542
15	29500	5028.7000

边界条件为

$y_1' = 0.4770\times10^{-1}, y_n' = 0.8810\times10^{-3}$

请编程序上机用三弯矩方程求三次样条函数第一类定解问题的解,

求 x_0=-900 开始，每间隔 Δx=1000,一共 n_0=31 个点上的函数值。单位毫米，当插值点 x 落在 [1162.5, 29500]外面时，规定节点处的 y 值取大数 10^7。

```
#include <cstdio>
#include <cmath>
#define MAX 10000000
#define N 15
double x[]={1162.5,2950,5900,7675.8515,8850,10881.255,11800,
13802.899,14750,17467.481,17700,20650,23600,26550,29500};

double y[]={99.5,220.2355,589.83954,950,1260.2342,1900,
2207.0194,2850,3124.1306,3800,3853.2175,4390.3286,4756.1508,
4976.2542,5028.7000};
```

```
int WINTH;
double dy0=0.04770,dyn=0.008810;
double M[N];
double g[N],u[N],v[N],t[N];
double h[N];

void cal(void)
{
	int i;
	for(i=0;i<N;i++) M[i]=g[i];
	v[0]/=t[0];
	for(i=1;i<N-1;i++) {
		t[i]-=u[i-1]*v[i-1];
		v[i]/=t[i];
	}
	t[N-1]-=u[N-2]*v[N-2];
	M[0]/=t[0];
	for(i=1;i<N;i++) M[i]=(M[i]-u[i-1]*M[i-1])/t[i];
	for(i=N-2;i>=0;i--) M[i]-=v[i]*M[i+1];
}

int get_n(double temp)
{
	int i;
	if(temp<x[0]) {
		return 0;
	} else if (temp>x[N-1]) {
		return N;
	} else {
		for(i=1;i<N;i++)
			if(temp>=x[i-1]&&temp<=x[i])
				return i;
	}
	return 0;
}

double spline(double k)
{
	int i;
	i=get_n(k);
	if(i<1||i>=N) return MAX;
	else {
		k=M[i-1]*(x[i]-k)*(x[i]-k)*(x[i]-k)/(6.0*h[i-1])
		+M[i]*(k-x[i-1])*(k-x[i-1])*(k-x[i-1])/(6.0*h[i-1])
```

```
            +(y[i-1]-M[i-1]*h[i-1]*h[i-1]/6.0)*(x[i]-k)/h[i-1]
            +(y[i]-M[i]*h[i-1]*h[i-1]/6.0)*(k-x[i-1])/h[i-1];
            return k;
        }
}
void output(void)
{
        double x0;
        for(x0=-900;x0<=29500;x0+=WINTH) {
            printf("x0=%.2lf,      spline(x0)=%-16.2e\n",x0,spline(x0));
        }
}
void main()
{
        int i;
        printf("please input delta X :\n");
        scanf("%d",&WINTH);
        for(i=0;i<N;i++) t[i]=2;
        for(i=0;i<N-1;i++) h[i]=x[i+1]-x[i];

        for(i=0;i<N-2;i++) u[i]=h[i]/(h[i]+h[i+1]);
        u[N-2]=1.0;

        for(i=1;i<N-1;i++) v[i]=1-u[i-1];
        v[0]=1.0;

        for(i=1;i<N-1;i++)
            g[i]=6/(h[i-1]+h[i])*((y[i+1]-y[i])/h[i]
            -(y[i]-y[i-1])/h[i-1]);
        g[0]=6.0*((y[1]-y[0])/h[0]-dy0)/h[0];
        g[N-1]=6.0*(dyn-(y[N-1]-y[N-2])/h[N-2])/h[N-2];

        cal();
        output();
}
```

运行：

please input delta X :
1000
x0=-900.00,   spline(x0)=1.00e+007
x0=1000.00,   spline(x0)=1.00e+007
x0=1100.00,   spline(x0)=1.00e+007
x0=2100.00,   spline(x0)=1.54e+002

x0=3100.00, spline(x0)=2.34e+002
x0=4100.00, spline(x0)=3.36e+002
x0=5100.00, spline(x0)=4.65e+002
x0=6100.00, spline(x0)=6.24e+002
x0=7100.00, spline(x0)=8.19e+002
x0=8100.00, spline(x0)=1.06e+003
x0=9100.00, spline(x0)=1.33e+003
x0=10100.00, spline(x0)=1.64e+003
x0=11100.00, spline(x0)=1.97e+003
x0=12100.00, spline(x0)=2.31e+003
x0=13100.00, spline(x0)=2.63e+003
x0=14100.00, spline(x0)=2.94e+003
x0=15100.00, spline(x0)=3.22e+003
x0=16100.00, spline(x0)=3.48e+003
x0=17100.00, spline(x0)=3.71e+003
x0=18100.00, spline(x0)=3.94e+003
x0=19100.00, spline(x0)=4.14e+003
x0=20100.00, spline(x0)=4.31e+003
x0=21100.00, spline(x0)=4.46e+003
x0=22100.00, spline(x0)=4.59e+003
x0=23100.00, spline(x0)=4.70e+003
x0=24100.00, spline(x0)=4.81e+003
x0=25100.00, spline(x0)=4.89e+003
x0=26100.00, spline(x0)=4.95e+003
x0=27100.00, spline(x0)=5.00e+003
x0=28100.00, spline(x0)=5.02e+003
x0=29100.00, spline(x0)=5.03e+003

第七章

习题 7-2

2.对 y=sinx 在[0, 90]，取 1 度为间隔，分别用库函数 sin(x)和用正交多项式对 sinx 做四次曲线拟合，比较拟合的结果与库函数的计算结果，并给出拟合曲线的系数.

```
#include<cstdio>
#include<cstdlib>
#include<cmath>

#define N 91
#define NUM 10
```

```
#define O 3.1415926/180
#define K 4

double delta(double s[],int ns,double x[],double y[],int m);
double Poly(double p[],int n,double x0)
{
	register int k;
	double f;
	f=p[n];
	for(k=n-1;k>=0;k--) {
		f=f*x0+p[k];
	}
	return f;
}

void mulpoly(double p1[],int n1,double p2[],int n2,double p[],int *n)
{
	int i,j;
	*n=n1+n2;
	for(i=0;i<=n1+n2;i++) p[i]=0;
	for(i=0;i<=n1;i++) {
		for(j=0;j<=n2;j++) {
			p[i+j]+=p1[i]*p2[j];
		}
	}
}

void addpoly(double p1[],int n1,double c1,double p2[],int n2,double c2,double p[],int *n)
{
	int i;
	int min;
	*n=(n1>=n2)?n1:n2;
	min=(n1<=n2)?n1:n2;
	for(i=0;i<=min;i++) {
		p[i]=p1[i]*c1+p2[i]*c2;
	}
	for(i=min+1;i<=n1;i++) {
		p[i]=p1[i]*c1;
	}
	for(i=min+1;i<=n2;i++) {
		p[i]=p2[i]*c2;
	}
}
void zj(double x[],double y[],int m)
```

```
{
    double sum,sum1,result[N];
    double p[NUM][NUM],temppoly[NUM],a[NUM],temppoly1[2];
    int n[NUM],ntemp,nr;
    int i,k;
    double alf[NUM],beta[NUM],temp,temp1,eps=0.00001;
    n[0]=0;
    p[0][0]=1;
    for(sum=0,i=0;i<m;i++) {
        sum+=y[i];
    }
    a[0]=sum/m;
    for(sum=0,i=0;i<m;i++) {
        sum+=x[i];
    }
    alf[1]=sum/m;
    n[1]=1;
    p[1][1]=1;
    p[1][0]=-alf[1];
    for(sum=0,i=0;i<m;i++) {
        sum+=y[i]*(x[i]-alf[1]);
    }
    for(sum1=0,i=0;i<m;i++) {
        sum1+=(x[i]-alf[1])*(x[i]-alf[1]);
    }
    a[1]=sum/sum1;
    addpoly(p[0],n[0],a[0],p[1],n[1],a[1],result,&nr);
    k=1;
    do {
        k++;
        for(sum=0,i=0;i<m;i++) {
            temp=Poly(p[k-1],n[k-1],x[i]);
            sum+=x[i]*temp*temp;
        }
        for(sum1=0,i=0;i<m;i++) {
            temp=Poly(p[k-1],n[k-1],x[i]);
            sum1+=temp*temp;
        }
        alf[k]=sum/sum1;
        for(sum=0,i=0;i<m;i++) {
            temp=Poly(p[k-1],n[k-1],x[i]);
            sum+=temp*temp;
        }
        for(sum1=0,i=0;i<m;i++) {
```

```
            temp1=Poly(p[k-2],n[k-2],x[i]);
            sum1+=temp1*temp1;
        }
        beta[k-1]=sum/sum1;
        temppoly1[1]=1;
        temppoly1[0]=-alf[k];
        mulpoly(temppoly1,1,p[k-1],n[k-1],temppoly,&ntemp);
        addpoly(temppoly,ntemp,1,p[k-2],n[k-2],-beta[k-1],p[k],&n[k]);
        for(sum=0,i=0;i<m;i++) {
            sum+=y[i]*Poly(p[k],n[k],x[i]);
        }
        for(sum1=0,i=0;i<m;i++) {
            temp=Poly(p[k],n[k],x[i]);
            sum1+=temp*temp;
        }
        a[k]=sum/sum1;
        for(i=0;i<=nr;i++) temppoly[i]=result[i];
        addpoly(temppoly,nr,1,p[k],n[k],a[k],result,&nr);
    } while(delta(result,nr,x,y,m)>=eps&&k<m-1);
    for(i=0;i<N;i++) {
        printf("\nsin(%.2d) = %.4f        poly(%.2d)=%.4f",i,sin(i*O),i,Poly(result,nr,i));
    }
    printf("\n");
    for(i=0;i<=nr;i++) {
        printf("c%d = %e\n ",i,result[i]);
    }
    printf("\n");
}
double delta(double s[],int ns,double x[],double y[],int m)
{
    int i;
    double sum,temp;

    for(sum=0,i=0;i<m;i++) {
        temp=Poly(s,ns,x[i]);
        temp-=y[i];
        sum+=temp*temp;
    }
    return sum;
}
void main(void)
{
    int i;
    double y[N],x[N];
```

```
    for(i=0;i<N;i++) {
        y[i]=sin(i*O);
        x[i]=i;
    }
    zj(x,y,N);
}
```

运行:

```
sin(00) = 0.0000        poly(00)=0.0002
sin(01) = 0.0175        poly(01)=0.0176
sin(02) = 0.0349        poly(02)=0.0350
sin(03) = 0.0523        poly(03)=0.0524
sin(04) = 0.0698        poly(04)=0.0697
sin(05) = 0.0872        poly(05)=0.0871
sin(06) = 0.1045        poly(06)=0.1045
sin(07) = 0.1219        poly(07)=0.1218
sin(08) = 0.1392        poly(08)=0.1391
sin(09) = 0.1564        poly(09)=0.1563
sin(10) = 0.1736        poly(10)=0.1735
sin(11) = 0.1908        poly(11)=0.1907
sin(12) = 0.2079        poly(12)=0.2078
sin(13) = 0.2250        poly(13)=0.2249
sin(14) = 0.2419        poly(14)=0.2418
sin(15) = 0.2588        poly(15)=0.2588
sin(16) = 0.2756        poly(16)=0.2756
sin(17) = 0.2924        poly(17)=0.2923
sin(18) = 0.3090        poly(18)=0.3090
sin(19) = 0.3256        poly(19)=0.3256
sin(20) = 0.3420        poly(20)=0.3420
sin(21) = 0.3584        poly(21)=0.3584
sin(22) = 0.3746        poly(22)=0.3746
sin(23) = 0.3907        poly(23)=0.3908
sin(24) = 0.4067        poly(24)=0.4068
sin(25) = 0.4226        poly(25)=0.4227
sin(26) = 0.4384        poly(26)=0.4384
sin(27) = 0.4540        poly(27)=0.4541
sin(28) = 0.4695        poly(28)=0.4695
sin(29) = 0.4848        poly(29)=0.4849
sin(30) = 0.5000        poly(30)=0.5001
sin(31) = 0.5150        poly(31)=0.5151
sin(32) = 0.5299        poly(32)=0.5300
sin(33) = 0.5446        poly(33)=0.5447
sin(34) = 0.5592        poly(34)=0.5593
```

sin(35) = 0.5736 poly(35)=0.5737
sin(36) = 0.5878 poly(36)=0.5879
sin(37) = 0.6018 poly(37)=0.6019
sin(38) = 0.6157 poly(38)=0.6157
sin(39) = 0.6293 poly(39)=0.6294
sin(40) = 0.6428 poly(40)=0.6428
sin(41) = 0.6561 poly(41)=0.6561
sin(42) = 0.6691 poly(42)=0.6692
sin(43) = 0.6820 poly(43)=0.6820
sin(44) = 0.6947 poly(44)=0.6947
sin(45) = 0.7071 poly(45)=0.7071
sin(46) = 0.7193 poly(46)=0.7193
sin(47) = 0.7314 poly(47)=0.7313
sin(48) = 0.7431 poly(48)=0.7431
sin(49) = 0.7547 poly(49)=0.7547
sin(50) = 0.7660 poly(50)=0.7660
sin(51) = 0.7771 poly(51)=0.7771
sin(52) = 0.7880 poly(52)=0.7879
sin(53) = 0.7986 poly(53)=0.7986
sin(54) = 0.8090 poly(54)=0.8089
sin(55) = 0.8192 poly(55)=0.8191
sin(56) = 0.8290 poly(56)=0.8290
sin(57) = 0.8387 poly(57)=0.8386
sin(58) = 0.8480 poly(58)=0.8480
sin(59) = 0.8572 poly(59)=0.8571
sin(60) = 0.8660 poly(60)=0.8660
sin(61) = 0.8746 poly(61)=0.8745
sin(62) = 0.8829 poly(62)=0.8829
sin(63) = 0.8910 poly(63)=0.8909
sin(64) = 0.8988 poly(64)=0.8987
sin(65) = 0.9063 poly(65)=0.9063
sin(66) = 0.9135 poly(66)=0.9135
sin(67) = 0.9205 poly(67)=0.9205
sin(68) = 0.9272 poly(68)=0.9272
sin(69) = 0.9336 poly(69)=0.9336
sin(70) = 0.9397 poly(70)=0.9397
sin(71) = 0.9455 poly(71)=0.9455
sin(72) = 0.9511 poly(72)=0.9511
sin(73) = 0.9563 poly(73)=0.9564
sin(74) = 0.9613 poly(74)=0.9613
sin(75) = 0.9659 poly(75)=0.9660
sin(76) = 0.9703 poly(76)=0.9704
sin(77) = 0.9744 poly(77)=0.9745
sin(78) = 0.9781 poly(78)=0.9782

sin(79) = 0.9816 poly(79)=0.9817
sin(80) = 0.9848 poly(80)=0.9849
sin(81) = 0.9877 poly(81)=0.9878
sin(82) = 0.9903 poly(82)=0.9903
sin(83) = 0.9925 poly(83)=0.9926
sin(84) = 0.9945 poly(84)=0.9946
sin(85) = 0.9962 poly(85)=0.9962
sin(86) = 0.9976 poly(86)=0.9976
sin(87) = 0.9986 poly(87)=0.9986
sin(88) = 0.9994 poly(88)=0.9993
sin(89) = 0.9998 poly(89)=0.9997
sin(90) = 1.0000 poly(90)=0.9998

c0 = 2.053431e-004
c1 =1.737655e-002
c2 = 6.299221e-006
c3 = -1.083126e-006
c4 = 2.656575e-009

第八章

习题 8-2

5. 用复化梯形公式求积分$\int_0^1 \frac{1}{1+\sin^2 x}$， 使误差不超过 10^{-5}。

```
#include <stdio.h>
#include <math.h>

double ciff(double x)
{
    return 1.0/(1.0+sin(x)*sin(x));
}

double cif(double a,double b,double eps)
{
    int n,k;
    double fa,fb,h,t1,p,s,x,t;

    fa=ciff(a); fb=ciff(b);
    n=1; h=b-a;
    t1=h*(fa+fb)/2.0;
    p=eps+1.0;
```

```
    while (p>=eps) {
        s=0.0;
        for (k=0;k<=n-1;k++) {
            x=a+(k+0.5)*h;
            s=s+ciff(x);
        }
        t=(t1+h*s)/2.0;
        p=fabs(t1-t);
        t1=t; n=n+n; h=h/2.0;
    }
    return t;
}

int main()
{
    double a,b,eps,t;

    printf("a=");
    scanf("%lf",&a);
    printf("b=");
    scanf("%lf",&b);
    printf("eps=");
    scanf("%lf",&eps);

    t=cif(a,b,eps);

    printf("\nt=%.3f\n",t);
    return 0;
}
```

运行：

a=0
b=1
eps=1e-5
t=0.809

习题 8-3

1.用龙贝格积分算法求积分 $\int_1^3 \frac{1}{y}dy$，**要求** $\left|T_k^{(0)} - T_{k-1}^{(0)}\right| < 10^{-5}$ 。

```
#include <cstdio>
#include <cmath>
```

```
double f(double x)
{
	return 1.0/x;
}

double romb(double a,double b,double eps)
{
	double rombf();
	int m,n,i,k;
	double y[10],h,ep,p,x,s,q;

	h=b-a;
	y[0]=h*(f(a) + f(b))/2.0;
	m=1; n=1; ep=eps+1.0;

	while ((ep>=eps)&&(m<=9)) {
		p=0.0;
		for (i=0;i<=n-1;i++) {
			x=a+(i+0.5)*h;
			p=p+f(x);
		}
		p=(y[0]+h*p)/2.0;
		s=1.0;
		for (k=1;k<=m;k++) {
			s=4.0*s;
			q=(s*p-y[k-1])/(s-1.0);
			y[k-1]=p; p=q;
		}
		ep=fabs(q-y[m-1]);
		m=m+1; y[m-1]=q; n=n+n; h=h/2.0;
	}
	return q;
}

int main()
{
	double a,b,eps,t;

	printf("a=");
	scanf("%lf",&a);
	printf("b=");
	scanf("%lf",&b);
```

```
    printf("eps=");
    scanf("%lf",&eps);

    t=romb(a,b,eps);

    printf("\nt=%.3f\n",t);
    return 0;
}
```

运行：
a=1
b=3
eps=1e-5
t=1.099

第九章

习题 9-3

2. 上机编程，用有限差分法求解

$$\begin{cases} y'' = -(x+1)y' + 2y + (1-x^2)e^{-x} & x \in (0,1) \\ y(0) = -1, \quad y(1) = 0 \end{cases}$$

取$h = 0.05$,并将结果与精确解 $y = (x-1)e^{-x}$比较。

```
#include <cstdio>
#include <cmath>

#define N 1000
double a[N];
double b[N];
double c[N];
double f[N];
double result[N];

double r(double x) {
    return (1-x*x) * exp(0.5*(x*x*0.5+x) - x);
}
double q(double x) {
    return 2.5 + 0.25*(x+1)*(x+1);
}
double y(double vx, double x) {
```

```
    return vx * exp(-0.25*(x*x+2*x));
}

double func(double x) {
    return (x-1) * exp(-x);
}

int main() {
    int i, n;
    double h;
    printf("Input n:");
    scanf("%d", &n);
    h = 1.0 / n;

    a[0] = a[n] = 0;
    b[0] = b[n] = 1;
    c[0] = c[n] = 0;
    f[0] = -1;
    f[n] = 0;
    for (i = 1; i < n; i++) {
        a[i] = 1;
        b[i] = -(2+q(i*h)*h*h);
        c[i] = 1;
        f[i] = r(i*h)*h*h;
    }

    c[0] /= b[0];
    for (i = 1; i < n; i++) c[i] = c[i] / (b[i] - a[i]*c[i-1]);

    result[0] = f[0] / b[0];
    for (i = 1; i < n; i++) result[i] = (f[i]-a[i]*result[i-1])/(b[i]-a[i]*c[i-1]);
    for (i = n -1; i >= 0; i--) result[i] -= c[i]*result[i+1];

    for (i=0; i <= n; i++) {
        printf("y(%2d) = %lf          y(%lf) = %lf\n", i, y(result[i], i*h), i*h, func(i*h));
    }
    return 0;
}
```

运行：

Input n:
20
y(0) = -1.000000   y(0.000000) = -1.000000

y(1) = -0.903679　　y(0.050000) = -0.903668
y(2) = -0.814372　　y(0.100000) = -0.814354
y(3) = -0.731626　　y(0.150000) = -0.731602
y(4) = -0.655013　　y(0.200000) = -0.654985
y(5) = -0.584131　　y(0.250000) = -0.584101
y(6) = -0.518604　　y(0.300000) = -0.518573
y(7) = -0.458078　　y(0.350000) = -0.458047
y(8) = -0.402222　　y(0.400000) = -0.402192
y(9) = -0.350724　　y(0.450000) = -0.350695
y(10) = -0.303291　　y(0.500000) = -0.303265
y(11) = -0.259651　　y(0.550000) = -0.259627
y(12) = -0.219546　　y(0.600000) = -0.219525
y(13) = -0.182734　　y(0.650000) = -0.182716
y(14) = -0.148991　　y(0.700000) = -0.148976
y(15) = -0.118104　　y(0.750000) = -0.118092
y(16) = -0.089875　　y(0.800000) = -0.089866
y(17) = -0.064119　　y(0.850000) = -0.064112
y(18) = -0.040661　　y(0.900000) = -0.040657
y(19) = -0.019339　　y(0.950000) = -0.019337
y(20) = 0.000000　　y(1.000000) = 0.000000

第十一章

习题 11-1

4. 上机求

$A=\begin{bmatrix}1 & 2 & 3\\ 2 & 3 & 4\\ 3 & 4 & 5\end{bmatrix}$ 在 -0.6 附近的特征根以及对应 的特征向量。

解：用反幂法求 $(A+0.6I)^{-1}$的特征根$\frac{1}{\lambda_i-0.6}$,记为λ,

从 $\frac{1}{\lambda_i-0.6}=\lambda$ 中解出 λ_i 即为所求。

```
#include <cstdio>
#include <cstdlib>
#include <cmath>

#define P    (-0.6)
#define COUNT 100
```

```
#define DIMENSION 3

float a[DIMENSION][DIMENSION]={
        1, 2, 3,
        2, 3, 4,
        3, 4, 5
};
float v[DIMENSION]={1.0,1.0,1.0};
float u[DIMENSION]={1.0,1.0,1.0};
void creattable(float b[DIMENSION][DIMENSION]);
void doolittlemethod(float a[DIMENSION][DIMENSION],float u[DIMENSION]);
float powermethod(float v[DIMENSION],float a[DIMENSION][DIMENSION],
                float u[DIMENSION],float eps);

int main()
{
   float value;
   float tmp;
   int i,j,k,r;
   float temp1=0.0;
   float temp2=0.0;
   float eps;

   printf("Please input the eps: ");
   scanf("%f",&eps);
   creattable(a);
   for(r=0;r<DIMENSION;r++) {
         for(j=r;j<DIMENSION;j++) {
               temp1=0.0;
               for(k=0;k<r;k++) temp1+=a[r][k]*a[k][j];
               a[r][j]-=temp1;
         }
         for(i=r+1;i<DIMENSION;i++) {
               temp2=0.0;
               for(k=0;k<r;k++) temp2+=a[i][k]*a[k][r];
               a[i][r]=(a[i][r]-temp2)/a[r][r];
         }
   }

   tmp=powermethod(v,a,u,eps);

   if(tmp!=0.0) {
         value = 1.0/tmp+P;
         printf("The value is %.3f\n",value);
```

```
        printf("The vector is: [");
        for(i=0;i<DIMENSION;i++){
            if (i > 0) printf(" ,");
            printf("%.3f",u[i]);
        }
        printf("]\n");
    }
    return 0;
}

void creattable(float b[DIMENSION][DIMENSION])
{
    int k;
    for(k=0;k<DIMENSION;k++) b[k][k]-=P;
}

float powermethod(float v[DIMENSION],float a[DIMENSION][DIMENSION],
            float u[DIMENSION],float eps)
{
    static float lam=0.0;
    float max=0.0;
    int i,k;

    for(i=0;i<COUNT;i++) {
        max=0.0;
        for(k=0;k<DIMENSION;k++)
        if(fabs(v[k])>fabs(max)) max=v[k];
        if(fabs(lam-max)<=eps) break;
        lam=max;
        for(k=0;k<DIMENSION;k++) u[k]=v[k]/max;
        doolittlemethod(a,u);
        for(k=0;k<DIMENSION;k++) v[k]=u[k];
        if(i==COUNT-1) {
            printf("No concergence\n");
            exit(0);
        }
    }

    for(k=0;k<DIMENSION;k++) u[k]=v[k]/max;
    return max;
}

void doolittlemethod(float a[DIMENSION][DIMENSION],float u[DIMENSION])
{
```

```
    int i,k;
    float temp=0.0;

    for(i=0;i<DIMENSION;i++) {
        temp=0.0;
        for(k=0;k<i;k++) temp+=a[i][k]*u[k];
        u[i]-=temp;
    }
    for(i=DIMENSION-1;i>=0;i--)
    {
        temp=0.0;
        for(k=i+1;k<DIMENSION;k++) temp+=a[i][k]*u[k];
        u[i]=(u[i]-temp)/a[i][i];
    }
}
```

运行：

Please input the eps:

1e-4

The value is -0.623

The vector is: [1.000 ,0.172 ,-0.656]