附录-全部源代码

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

double ep = 1e-12, b=0.16,c=-0.064;
double p[501],l1[501],up1[501],up2[501];
double power(double a[]);
double inv_power(double a[]);
void LUde(double a[]);
double det(double a[]);

int main(){
    int i,k;
    double A[501],B[501],beta_1,beta_501,beta_s,beta_k;
    double mu;
    for(i=0;i<501;i++)
        A[i]=(1.64-0.024*(i+1))*sin(0.2*(i+1))-0.64*exp(0.1/(i+1));
    beta_1=power(A);
    printf("λ1\t= %.12e\n",beta_1);
    for(i=0;i<501;i++) //位移
     B[i]=A[i]-beta_1;
    beta_501=power(B)+beta_1;
    printf("λ501\t= %.12e\n",beta_501);
    beta_s=inv_power(A);
    printf("λs\t= %.12e\n",beta_s);
    for(k=1;k<=39;k++)
    {
        mu=beta_1+k*(beta_501-beta_1)/40;
        for(i=0;i<501;i++)
            B[i]=A[i]-mu;
        beta_k=inv_power(B)+mu;
        printf("λi%d\t= %.12e\n",k,beta_k);
    }
    printf("cond(A)2= %.12e\n",beta_1/beta_s);//求解条件数
    printf("detA\t= %.12e\n",det(A)); //求解特征值
 //   return 0;
}

double power(double a[]){
    int j,N = 5000;
    double b = 0.16, c = -0.064;
    double u[501], y[501];
    double m = 1, beta;
    for (int i = 0; i < 501; i++) {
        u[i] = 1;
    }
    j = 0;
    while (j<N){
        for (int i = 0; i < 501; i++) {
            y[i] = u[i]/fabs(m);
        }
        u[0]=a[0]*y[0]+b*y[1]+c*y[2];
        u[1]=b*y[0]+a[1]*y[1]+b*y[2]+c*y[3];
```

```
        u[499]=c*y[497]+b*y[498]+a[499]*y[499]+b*y[500];
        u[500]=c*y[498]+b*y[499]+a[500]*y[500];
        for (int i = 2;i<499;i++){
            u[i] = c*y[i-2]+b*y[i-1]+a[i]*y[i]+b*y[i+1]+c*y[i+2];
        }
        beta = 0;
        for (int i = 0;i<501;i++){
            if (fabs(u[i])>=fabs(beta))
                beta = u[i];
        }
        if(fabs(beta-m)/fabs(beta)<ep)
            break;
        if(beta<0)
            if(fabs(fabs(beta)-fabs(m))/fabs(beta)<ep)
                break;
        m = beta;
        j++;
    }
    return beta;

}
void LUde(double a[]){
    p[0]=a[0];
    l1[0]=b;
    up1[0]=b/p[0];
    up2[0]=c/p[0];
    p[1]=a[1]-l1[0]*up1[0];
    up2[1]=c/p[1];
    up1[1]=(b-l1[0]*up2[0])/p[1];
    l1[1] = b - c*up1[0];
    for(int i=2;i<501;i++)
    {
        l1[i]=b-c*up1[i-1];
        p[i]=a[i]-c*up2[i-2]-l1[i-1]*up1[i-1];
        up2[i]=c/p[i];
        up1[i]=(b-l1[i-1]*up2[i-1])/p[i];
    }
}

double inv_power(double a[]) //反幂法
{
    double u[501],y[501]; //LU
    double beta,m=1;
    int i,j,N=1000;
    LUde(a);
    for(i=0;i<501;i++)
        u[i]=1;
    j=0;
    while(j<N)
    {
        for(i=0;i<501;i++)
        {
            y[i]=u[i]/fabs(m);
        }
        u[0]=y[0]/p[0];
        u[1]=(y[1]-l1[0]*u[0])/p[1];
        for(i=2;i<501;i++)
            u[i]=(y[i]-c*u[i-2]-l1[i-1]*u[i-1])/p[i];
```

```
        u[499]=u[499]-up1[499]*u[500];
        for(i=498;i>=0;i--)
            u[i]=u[i]-up1[i]*u[i+1]-up2[i]*u[i+2];

        beta=0;
        for(i=0;i<501;i++)
        {
            if(fabs(u[i])>=fabs(beta))
                beta=u[i];
        }
        if(beta<0)
            if(fabs(fabs(beta)-fabs(m))/fabs(beta)<ep)
                break;
        if(fabs(beta-m)/fabs(beta)<ep)
            break;
        m=beta;
        j++;
    }
    return 1/beta;
}
double det(double a[]) //求det
{
    double det_A=1;
    LUde(a);
    for(int i=0;i<501;i++)
        det_A=det_A*p[i];
    return det_A;
}
```