Lab3 Orthogonal Polynomials Approximation

Sheng Zhang
2016310200210
2nd class of information and computing science

1 Lab purpose

Given m sample points $x_1 < x_2 < ... < x_m$, corresponding function value $f(x_i)$ and the weight of every sample point $w_i(1 <= i <= m)$ respectively. Please find the orthogonal polynomial of curve fitting, which satisfies the least square error.

$$||err||^2 = \sum_{i=1}^{m} = w_i[f(x_i) - P_n(x_i)]^2 < TOL \tag{1}$$

2 Lab Requirements

2.1 Definition of Function interface

```
int OPA( double (*f)(double t), int m, double x[], double w[], double c
    [], double *eps )
```

The original code is as follow:

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

#define MAXM 200
#define MAXN 5


double f1(double x)
{
  return sin(x);
}

double f2(double x)
{
  return exp(x);
}

int OPA( double (*f)(double t), int m, double x[], double w[], double c
    [], double *eps );

void print_results( int n, double c[], double eps)
{
  int i;

  printf("%d\n", n);
  for (i=0; i<=n; i++)
    printf("%8.4e ", c[i]);
  printf("\n");
  printf("error = %6.2e\n", eps);
  printf("\n");
}

int main()
{
```

```
  int m, i, n;
  double x[MAXM], w[MAXM], c[MAXN+1], eps;

  m = 90;
  for (i=0; i<m; i++) {
    x[i] = 3.1415926535897932 * (double)(i+1) / 180.0;
    w[i] = 1.0;
  }
  eps = 0.001;
  n = OPA(f1, m, x, w, c, &eps);
  print_results(n, c, eps);
  m = 200;
  for (i=0; i<m; i++) {
    x[i] = 0.01*(double)i;
    w[i] = 1.0;
  }
  eps = 0.001;
  n = OPA(f2, m, x, w, c, &eps);
  print_results(n, c, eps);
  return 0;
}
```

2.2 Data requirements

3
-2.5301e-003 1.0287e+000 -7.2279e-002 -1.1287e-001
error = 6.33e-005

4
1.0025e+000 9.6180e-001 6.2900e-001 7.0907e-003 1.1792e-001
error = 1.62e-004

3 Method

3.1 Detail of algorithm

To approximate a given function by a polynomial with error bounded by a given tolerance.
Input: number of data m; x[m]; y[m]; weight w[m]; tolerance TOL;maximum degree of polynomial MAXN.
Output: coefficients of the approximating polynomial.
Here is the detail of algorithm. According to given conditions, the expression can be represented easily.

$$\begin{aligned} err &= ||P-y||^2 \\ &= (P-y, P-y) \\ &= (\sum_{k=0}^{n} a_k\varphi_k - y, \sum_{i=0}^{n} a_i\varphi_i - y) \\ &= \sum_{k=0}^{n} a_k^2(\varphi_k,\varphi_k) - 2\sum_{k=0}^{n} a_k(\varphi_k, y) + (y,y) \\ &= (y,y) - \sum_{k=0}^{n} a_k(\varphi_k, y) \qquad (*) \end{aligned}$$

So the error can be computed by (*) easily.
Step 1:Set $\varphi_0(x) \equiv 1$;$a_0 = (\varphi_0, y)/(\varphi_0, \varphi_0)$; $err = (y,y) - a_0(\varphi_0, y)$
Step 2:Set $\alpha_1 = (x\varphi_0, \varphi_0)/(\varphi_0, \varphi_0)$; $\varphi_1(x) = (x-\alpha_1)\varphi_0(x)$; $a_1 = (\varphi_1, y)/(\varphi_1, \varphi_1)$; $err-= a_1(\varphi_1, y)$
Step 3:Set $k = 1$
Step 4:While((k<MAXN) && (|err|>=TOL) do steps 5-7
Step 5: k++;
step 6: $\alpha_k = (x\varphi_1, \varphi_1)/(\varphi_1, \varphi_1)$; $\beta_{k-1} = (\varphi_1, \varphi_1)/(\varphi_0, \varphi_0)$; $\varphi_2(x) = (x-\alpha_k)\varphi_1(x) - \beta_{k-1}\varphi_0(x)$;

$a_k = (\varphi_2, y)/(\varphi_2, \varphi_2)$; $err- = a_k(\varphi_2, y)$;
Step 7: Set $\varphi_0(x) = \varphi_1(x)$; $\varphi_1(x) = \varphi_2(x)$;
Step 8: Output(); STOP.

3.2 How to represent polynomials in computer

As we all know, polynomials can not be represented normally in computer memory. However, utilizing one way array, coefficients of the approximating polynomial can be stored easily.For instance, we can represent the polynomial $y = 3x^2+2x+1$ by one way array a[3] and correspondingly, $a[0] = 1, a[1] = 2, a[2] = 3$. I apply this method into my code, and it works very well.

3.3 How to compute polynomials in computer

Having understand the method of how to represent the polynomials in computer memory, how to compute polynomials is converted to how to compute the coefficients. In my code, phi0[MAXN+1], phi1[MAXN+1], phi2[MAXN+1] are used to store the coefficients of $\varphi_0(x)$, $\varphi_1(x)$, $\varphi_2(x)$. When I need certain $\varphi 2(x)$, I just only use pow(x,0)*phi2[0]+pow(x,1)*phi2[1]+...+pow(x,MAXN)*phi2[MAXN+1] to represent it. Take $\varphi_1(x) = (x-\alpha_1)\varphi_0(x)$ as an example, To finish this operation, I set phi1[1]=1 and phi1[0]=$-\alpha_1$ because here $\varphi_0(x) = 1$. Certainly, we can also finish other operations in a similar way.

4 Code analysis and thinking

I spite the implement of algorithm into four main parts, namely definition, step1, step2, and the remaining parts. Clearly, for every part, there will be no difficulty once I understand the meaning of symbols, then I only need to type the code according to the formula.
Sometimes, we know how to operate on the draft clearly, but we don't know how to do it on a computer. It's usual for a beginner. There are more complex problem than computing polynomials such as large integer problem. After getting a lot of exercise and mastering skills of programming, we understand how to solve similar problems normally.

5 Code sharing

The result is showed in Figure 1, and the whole code is also pasted below.

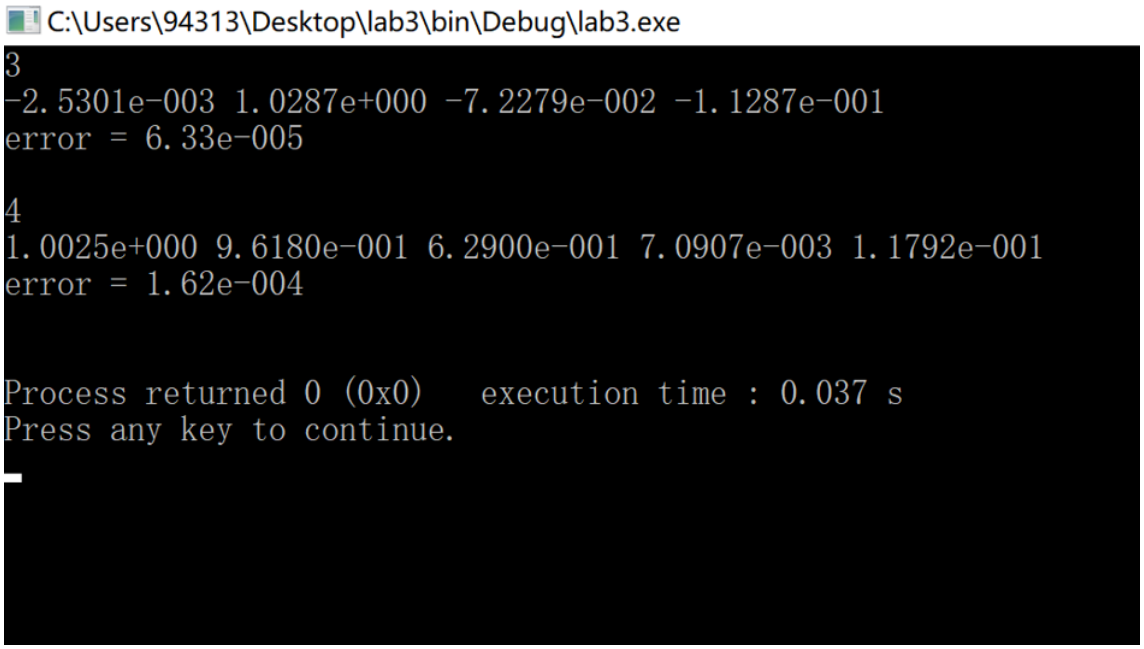

Figure 1: The result of code.

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

#define MAXM 200
#define MAXN 5


double f1(double x)
{
  return sin(x);
}

double f2(double x)
{
  return exp(x);
}

int OPA( double (*f)(double t), int m, double x[], double w[], double c
    [], double *eps )
{
    //----------------definition----------------
    double err;
    double phi0[MAXN+1],phi1[MAXN+1],phi2[MAXN+1];
    double alpha, beta;
    double a[MAXN+1];
    for(int i=0; i<=MAXN;i++)
    {
        phi0[i] = 0;
        phi1[i] = 0;
        phi2[i] = 0;
    }

    //----------------Step 1----------------------//

    phi0[0] = 1;
    double y_y=0, phi0_y=0, phi0_phi0=0;
    for(int i=0;i<m;i++)
    {
        phi0_y += w[i] * f(x[i]);
        phi0_phi0 += w[i];
    }
    a[0] = phi0_y / phi0_phi0;
    for(int i=0;i<m;i++)
        y_y += w[i] * f(x[i]) * f(x[i]);
    err = y_y - a[0] * phi0_y;
    for(int i=0; i<=MAXN; i++)
        c[i] = a[0] * phi0[i];

    //----------------Step 2----------------------//

    double xphi0_phi0=0, phi1_y=0, phi1_phi1=0;
    phi0_phi0 = 0;
    for(int i=0; i<m; i++)
    {
        xphi0_phi0 += w[i] * x[i];
        phi0_phi0 += w[i];
    }
    alpha = xphi0_phi0 / phi0_phi0;
    phi1[0] = -alpha;
    phi1[1] = 1;
    for(int i=0; i<m; i++)
    {
```

```
        phi1_y += w[i] * f(x[i]) * (x[i]-alpha);
        phi1_phi1 += w[i]* pow((x[i]-alpha),2);
    }
    a[1] = phi1_y / phi1_phi1;
    err -= a[1] * phi1_y;
    for(int i=0; i<=MAXN; i++)
        c[i] += a[1] * phi1[i];

    //----------------Step 345678--------------------//

    int k = 1;
    while( (k<MAXN) && (fabs(err)>=(*eps)))
    {
        k++;
        double xphi1_phi1=0;
        phi1_phi1 = 0, phi0_phi0 = 0;
        for(int i=0; i<m; i++)
        {
            double Phi0 = 0, Phi1 = 0;
            for(int j=0; j<=MAXN; j++)
            {
                Phi0 += phi0[j] * pow(x[i],j);
                Phi1 += phi1[j] * pow(x[i],j);
            }
            phi1_phi1 += w[i] * Phi1 * Phi1;
            xphi1_phi1 += w[i] * x[i] * Phi1 * Phi1;
            phi0_phi0 += w[i] * Phi0 * Phi0;
        }
        alpha = xphi1_phi1 / phi1_phi1;
        beta = phi1_phi1 / phi0_phi0;
        phi2[0] = -alpha*phi1[0] - beta*phi0[0];
        for(int i=1; i<=MAXN; i++)
            phi2[i] = phi1[i-1] - alpha*phi1[i] - beta*phi0[i];

        double phi2_y=0, phi2_phi2=0;
        for(int i=0;i<m;i++)
        {
            double Phi2=0;
            for(int j=0; j<=MAXN; j++)
                Phi2 += phi2[j] * pow(x[i],j);
            phi2_y += w[i]* Phi2 * f(x[i]);
            phi2_phi2 += w[i] * Phi2 * Phi2;
        }
        a[k] = phi2_y / phi2_phi2;
        err -= a[k] * phi2_y;
        for(int i=0; i<=MAXN; i++)
        {
            c[i] += a[k] * phi2[i];
            phi0[i] = phi1[i];
            phi1[i] = phi2[i];
        }
    }
    *eps = err;
    return k;
}
void print_results( int n, double c[], double eps)
{
  int i;

  printf("%d\n", n);
  for (i=0; i<=n; i++)
    printf("%8.4e ", c[i]);
  printf("\n");
```

```
  printf("error = %6.2e\n", eps);
  printf("\n");
}

int main()
{
  int m, i, n;
  double x[MAXM], w[MAXM], c[MAXN+1], eps;

  m = 90;
  for (i=0; i<m; i++) {
    x[i] = 3.1415926535897932 * (double)(i+1) / 180.0;
    w[i] = 1.0;
  }
  eps = 0.001;
  n = OPA(f1, m, x, w, c, &eps);
  print_results(n, c, eps);

  m = 200;
  for (i=0; i<m; i++) {
    x[i] = 0.01*(double)i;
    w[i] = 1.0;
  }
  eps = 0.001;
  n = OPA(f2, m, x, w, c, &eps);
  print_results(n, c, eps);

  return 0;
}
```