数値計算 Class-6 演習

21T2166D 渡辺大樹

2023/06/26

1　演習内容

Class-6 では伴って変化する二つの変数 x,y についてその変化の関係を調べるため、実験、観測などで得たいくつかの x,y の値を元にして x,y の関係を推定する補間法について C で実装する。

今回扱う補間法はラグランジュの補間多項式とニュートンの差商公式で、この 2 つについて以下に示していく。

1.0.1　ラグランジュの補間多項式

ラグランジュの補完法多項式ソースコード 1 で実装される。

ソースコード 1　laghkn.c

```
/***********************************************************************/
/* ラグランジュの補間多項式 laghkn.c */
/***********************************************************************/
#include <stdio.h>
#define N 11
    int
    main(void)
{
    int i, j, k, n, np;
    double seki, xx, s, x[N], y[N], dx;
    char z, zz;
    while (1)
    {
        printf("ラグランジュの補間多項式 \n");
        printf("補間点の個数を入力してください (1 < n < 10) n = ");
        scanf("%d%c", &n, &zz);
        if ((n <= 1) || (10 <= n))
            continue;
        printf("\n 補間点の座標を入力してください。\n");
```

```c
        for (i = 1; i <= n; i++)
        {
            printf(" x(%d) = ", i);
            scanf("%lf%c", &x[i], &zz);
            printf(" y(%d) = ", i);
            scanf("%lf%c", &y[i], &zz);
            printf("\n");
        }
        printf("\n 正し入力しましたか？ (y/n) ");
        scanf("%c%c", &z, &zz);
        if (z == 'y')
            break;
    }
    printf("\n 指定する点数は ？ np = ");
    scanf("%d%c", &np, &zz);
    dx = (x[n] - x[1]) / np;
    xx = x[1];
    for (i = 0; i <= np; i++)
    {
        s = 0.0;
        /*** ^^e2^^88^^91 Lk(x) の計算 ***/
        for (k = 1; k <= n; k++)
        {
            seki = 1.0;
            /*** Lk(x) の計算 ***/
            for (j = 1; j <= n; j++)
            {
                if (j != k)
                {
                    seki *= (xx - x[j]) / (x[k] - x[j]);
                }
            }
            s += seki * y[k];
        }
        printf("%10.6lf, %10.6lf \n", xx, s);
        xx += dx;
    }
    return 0;
}
```

このコードの動作をラグランジュの補間多項式とともに解説していく。

今 $x = x_1$ のとき $y = y_1$、$x = x_2$ のとき $y = y_2$、$x = x_3$ のとき $y = y_3$ であるような定数

$x_1, x_2, x_3, y_1, y_2, y_3$ を考える。

まず定数を 2 つに絞って考える。$x = x_1$ のとき $y = y_1$ でありたいので x についての一次式 $y = \frac{x-x_2}{x_1-x_2}y_1$ のような式を考えると $x = x_1$ のとき $y = y_1$、$x = x_2$ のとき $y = 0$ となる。同じように $x = x_2$ のとき $y = y_2$ になるような x の一次式を考えると $y = \frac{x-x_1}{x_2-x_1}y_2$ となり、$x = x_1$ のとき $y = 0$、$x = x_2$ のとき $y = y_2$ となる。

この二式を足し合わせることで

$$y = \frac{x-x_2}{x_1-x_2}y_1 + \frac{x-x_1}{x_2-x_1}y_2$$

という式が求まる。この式は 2 点 $(x_1, y_1), (x_2, y_2)$ を通る一次の直線を表している。

ここから 3 点 $x_1, x_2, x_3, y_1, y_2, y_3$ について考えていく。$x = x_1$ のとき $y = y_1$ でありたいので x についての一次式 $y = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}y_1$ のような式を考えると $x = x_1$ のとき $y = y_1$、$x = x_2, x_3$ のとき $y = 0$ となる。同様に考えると 3 式の和を考えて

$$y = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}y_1 + \frac{(x-x_3)(x-x_1)}{(x_2-x_3)(x_2-x_1)}y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}y_3$$

の式を得られる。これは $x = x_1$ のとき $y = y_1$、$x = x_2$ のとき $y = y_2$、$x = x_3$ のとき $y = y_3$ になっている。

したがってこの式は 3 点 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ を通る高々 2 次式を表す。

この式はデータの個数が増えても同様な式で表すことができて、データの個数を n+1、点の組み合わせを $(x_j, y_j), (j = 0, 1, 2, \cdots, n)$ で表す。まず y_k の係数となる部分を $L_k(x)$ とすると

$$L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j} (k = 0, 1, \cdots, n)$$

と表すことができる。

これを用いると $(x_j, y_j), (j = 0, 1, 2, \cdots, n)$ を通る n 次の多項式 $L(x)$ は

$$L(x) = \sum_{k=0}^{n} L_k(x) \cdot y_k$$

と表せる。

これがラグランジュの補間多項式となる。

この処理が実際にソースコード 1 に実装されている。具体的にこの計算が実装されているのは 40-53 行目であり、この for 文の中が $L(k)$ の足し合わせの計算になっており 45 行目からの for 文が $L_k(x)$ の掛け合わせの計算になっている。

このコードでは、多項式を出力するのではなくデータの最大値と最小値の間で、予測された多項式の出力したい点の個数を出力出来るようにしている。

1.1 ニュートンの差商公式

ニュートンの差商公式はソースコード 2 で実装される。

ソースコード 2　newton.c

```
#include <stdio.h>
#define N 10
    int
    main(void)
{
    int i, j, n;
    double a[N][N], s, t, x;
    char z, zz;
    while (1)
    {
        printf("ニュートンの差商公式による補間\n");
        printf("補間の個数 n は？ (1<n<9) n=");
        scanf("%d%c", &n, &zz);
        if ((n <= 1) || (9 <= n))
            continue;
        printf("\n補間点の座標を入力してください\n");
        for (i = 0; i < n; i++)
        {
            printf(" x(%d)=", i);
            scanf("%lf%c", &a[i][0], &zz);
            printf(" y(%d)=", i);
            scanf("%lf%c", &a[i][1], &zz);
        }
        printf("\n正しく入力しましたか? (y/n)");
        scanf("%c%c", &z, &zz);
        if (z == 'y')
            break;
    }
    /*** 各階差商の計算 ***/
    /*** 第２階差商をa[i][2]へ入れる***/
    /*** 第３階差商をa[i][3]へ入れる***/
    for (j = 1; j <= n; j++)
    {
        for (i = 0; i <= n - j; i++)
        {
            a[i][j + 1] = (a[i + 1][j] - a[i][j]) / (a[i + j][0] - a[i][0]);
```

```
        }
    }
    while (1)
    {
        printf("指定する点は？ X= ");
        scanf("%lf%c", &x, &zz);
        s = a[0][1];
        t = 1;
        /*** 差商公式による計算 ***/
        for (j = 2; j <= n; j++)
        {
            t *= (x - a[j - 2][0]);
            s += a[0][j] * t;
        }
        /*** 答の表示 ***/
        printf("\n f(%10.6lf) = %10.6lf\n", x, s);
        printf("\n やめますか？ (y/n) ");
        scanf("%c%c", &z, &zz);
        if (z == 'y')
            break;
    }
    return 0;
}
```

このコードを実際のニュートンの差商公式と比較しながら動作を確認していく。

関数 $f(x)$ について x の補間点 $x = x_0, x_1, \cdots, x_n$ における関数の値を $f_0, f_1, \cdots, f_n$ と表す。このときの関数 $f(x)$ の近似値を求めていく。

変数 x が点 x から x_0 まで変化するときの関数 $f(x)$ の平均変化率を $f[x, x_0]$ で表すと

$$f[x, x_0] = \frac{f(x) - f_0}{x - x_0}$$

となる。これを点 x, x_0 に関する $f(x)$ の第 1 階差商と呼ぶ。

式を変形させると

$$f(x) = f_0 + (x - x_0)f[x, x_0]$$

となる。

次に補間点 x_1 を追加する。前式の平均変化率 $f[x, x_0]$ を補間点 x_0 から x_1 の平均変化率 $f[x_0, x_1]$ で表したい。$f[x, x_0]$ は x の関数なので、x から x_1 までの $f[x, x_1]$ の平均変化率を $f[x, x_0, x_1]$ で表すと

$$f[x, x_0, x_1] = \frac{f[x, x_0] - f[x_0, x_1]}{x - x_1}$$

となる。これはいわば $f(x)$ の 2 次の平均変化率である。これを $f(x)$ の第 2 階差商という。

この式は変形させると

$$f[x, x_0] = f[x_0, x_1] + (x - x_1) f[x, x_0, x_1]$$

となり、第 1 階差商の変形式に代入すれば

$$f(x) = f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x, x_0, x_1]$$

となる。この式の x を定めると計算可能である部分とそうでない部分を分けると

$$\begin{cases} f(x) & \simeq f_0 + (x - x_0) f[x_0, x_1] \\ R_1 & = (x - x_0)(x - x_1) f[x, x_0, x_1] \end{cases}$$

となる。計算可能な部分が $f(x)$ の近似値となり、計算が不可能な部分 R_1 が誤差となる。

これを n で一般化すると、異なる $n + 1$ 個の補間点 $x = x_0, x_1, \cdots, x_n$ における関数の値を $f_0, f_1, \cdots, f_n$ と表すとき、任意の x に対して次の等式が成り立つ。

$$\begin{cases} f(x) & = f_0 + \sum_{i=0}^{n} ([\prod_{k=0}^{i} (x - x_k)] f[x_0, \cdots, x_{i+1}]) + R_n \\ R_n & = [\prod_{k=0}^{n} (x - x_k)] f[x, x_0, \cdots, x_n] \end{cases}$$

これがニュートンの差商公式となる。

ソースコード 2 では、上記の一般化の式を実装しているわけではなく、第 n 階差商をそれぞれ求めて、これを解を出力するときに用いる。(2 32-38)

2　演習結果

ラグランジュの補間多項式でも、ニュートンの差商公式でも以下の表のような補間点を入力して、実行する。

x	1	3	4	2
$f(x)$	1	2	5	?

ソースコード 1 での実行結果は以下のようになった。

```
指定する点数は ？ np = 3
1.000000, 1.000000
2.000000, 0.666667
3.000000, 2.000000
4.000000, 5.000000
```

ということで、$x = 2$ での値は $f(x) = 0.666667$ という結果となった。

ソースコード 2 での実行結果は以下のようになった。

```
指定する点は？ X= 2

f( 2.000000) = 0.666667
```

ということで、$x = 2$ での値は $f(x) = 0.666667$ という結果となった。