2003年度　応用数学　第1回レポート

080121691　宮田　考史

2003年11月5日

■問題1

(1) IEEE方式の単精度で、丸めに0捨1入を使った場合のマシンイプシロンを求める

仮数部をt桁とすると、

$$1 = 0.\underbrace{10\cdots00}_{t\text{桁}} \times 2^1$$

マシンイプシロンをεとおくと、

$$\begin{aligned} 1+\varepsilon &= 0.\underbrace{10\cdots00}_{t\text{桁}}1 \times 2^1 \\ \therefore \varepsilon &= 0.\underbrace{00\cdots00}_{t\text{桁}}1 \times 2^1 \\ &= 0.\underbrace{00\cdots01}_{t\text{桁}} \end{aligned}$$

よって、10進数であらわせば、$\varepsilon = 2^{-t}$である。

単精度では$t = 24$であるから、$\varepsilon = 2^{-24} = 5.96 \times 10^{-8}$となる。

(2) 同じ条件で、倍精度に対するマシンイプシロンを求める

倍精度では$t = 53$であるから、$\varepsilon = 2^{-53} = 1.11 \times 10^{-16}$となる。

■問題 2　計算過程で桁落ちが生じるか判定し、桁落ちが生じる場合、式変形を行う

値が近い数どうしの減算であり、桁落ちが生じる可能性があるので、式変形を行う。

(1)

$$\begin{aligned}\sqrt{1+x}-\sqrt{1-x} &= \frac{\left(\sqrt{1+x}-\sqrt{1-x}\right)\left(\sqrt{1+x}+\sqrt{1-x}\right)}{\sqrt{1+x}+\sqrt{1-x}} \\ &= \frac{1+x-(1-x)}{\sqrt{1+x}+\sqrt{1-x}} \\ &= \frac{2x}{\sqrt{1+x}+\sqrt{1-x}}\end{aligned}$$

(2)

$$\sin(1+x)-\sin(1-x) = 2\sin x\cos 1$$

(3)

$$\begin{aligned}\sqrt[4]{1+x}-1 &= \frac{\left(\sqrt[4]{1+x}-1\right)\left(\sqrt[4]{1+x}+1\right)}{\sqrt[4]{1+x}+1} \\ &= \frac{\left(\sqrt{1+x}-1\right)\left(\sqrt{1+x}+1\right)}{\left(\sqrt[4]{1+x}+1\right)\left(\sqrt{1+x}+1\right)} \\ &= \frac{1+x-1}{\left(\sqrt[4]{1+x}+1\right)\left(\sqrt{1+x}+1\right)} \\ &= \frac{x}{\left(\sqrt[4]{1+x}+1\right)\left(\sqrt{1+x}+1\right)}\end{aligned}$$

■問題 3

(1) 正の数 a の平方根を Newton 法によって求める

Newton 法では、初期近似値を x_0 として、

$$f(x) \simeq f(x_0) + f'(x_0)\,(x-x_0)$$

と近似する。$f(x)=0$ となる x は、

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

である。ここで、$x_0 = x$ とおき、反復を繰り返すのが、Newton 法の手順である。

$x = \sqrt{a}$ を解にもつ方程式 $f(x) = 0$ を考えれば、$f(x) = x^2 - a$ とすればよいことが分かる。ただし、$x = -\sqrt{a}$ も解にもつので、反復動作から、求めようとしている解 $x = \sqrt{a}$ に近づくように、適切な初期値 x_0 を設定しなければならない。

$$\therefore \text{反復式} \begin{cases} x &= x_0 - \dfrac{x_0^2 - a}{2x_0} \\ x_0 &= x \end{cases}$$

(2) Newton 法によって、$\sqrt{2}$ の値を計算し、誤差の減少する様子を調べる

(3) 二分法によって、$\sqrt{2}$ の値を計算し、誤差の減少する様子を調べる

Newton 法では初期値を $x_0 = 2$ とし、二分法では初期区間を $[a_0, b_0] = [1, 2]$ とした。Newton 法、二分法を用いて計算した結果を Table.1 に示す。

Table. 1: $\sqrt{2}$ の計算

	Newton 法		二分法			
n	x_n	$f(x_n)$	a_n	b_n	c_n	$f(c_n)$
0	2.000000	2.000000	1.000000	2.000000	1.500000	0.250000
1	1.500000	0.250000	1.000000	1.500000	1.250000	-0.437500
2	1.416667	0.006944	1.250000	1.500000	1.375000	-0.109375
3	1.414216	0.000006	1.375000	1.500000	1.437500	0.066406
4	1.414214	0.000000	1.375000	1.437500	1.406250	-0.022461
5	1.414214	0.000000	1.406250	1.437500	1.421875	0.021729
6	1.414214	0.000000	1.406250	1.421875	1.414062	-0.000427
7	1.414214	0.000000	1.414062	1.421875	1.417969	0.010635
8	1.414214	0.000000	1.414062	1.417969	1.416016	0.005100
9	1.414214	0.000000	1.414062	1.416016	1.415039	0.002336
10	1.414214	0.000000	1.414062	1.415039	1.414551	0.000954

Table.1 より、Newton 法では、$n = 4$ 以降は変化せず、収束が速い。(x_n の項を見ると、真の値は 1.414213... であるから、正しい値に近づいている)
二分法も真の値に近づいているが、$n = 10$ のときでも、小数点以下 3 桁までの精度であり、比較することで、Newton 法の収束の早さが理解できる。c_n の項を見ると、3 ～ 4 反復に 1 桁の割合で真の値に近づいている。(収束率 1/2 で一次収束)

■参考　Newton 法と二分法を用いて $\sqrt{a}$ $(a > 0)$ を計算するプログラムを示した。a の値、Newton 法で用いる初期値 x_0、二分法で用いる初期区間の境界値 a_0、b_0 を入力し、計算結果を出力する。Excel を媒介として、TEX に取り込むことを想定し、それぞれの計算結果を、項目ごとにまとめて出力し、リダイレクトした。

```
 1 #include <stdio.h>
 2 #define TRY     10      /* 反復回数 */
 3
 4 double f(double a, double x)
 5 {
 6         return ((x * x) - a);
 7 }
 8
 9 double df(double x)
10 {
11         return (2.0 * x);
12 }
13
14 void newton(int number, double a, double x_0)
15 {
16         int     i;
17         double  x = x_0;
18
19         for (i = 0; i <= TRY; i++) {
20                 switch (number) {
21                         case 0:
22                                 printf("%f\n", x);
23                                 break;
24                         case 1:
25                                 printf("%f\n", f(a, x));
26                                 break;
27                         default:
28                                 printf("ERROR!\n");
29                 }
30                 x  = x_0 - (f(a, x_0) / df(x_0));
31                 x_0 = x;
32         }
33         printf("\n");
```

```
34 }
35
36 void bisection(int number, double a, double a_n, double b_n)
37 {
38         int     i;
39         double  c_n;
40
41         for (i = 0; i <= TRY; i++) {
42                 c_n = (a_n + b_n) / 2;
43                 switch (number) {
44                         case 0:
45                                 printf("%f\n", a_n);
46                                 break;
47                         case 1:
48                                 printf("%f\n", b_n);
49                                 break;
50                         case 2:
51                                 printf("%f\n", c_n);
52                                 break;
53                         case 3:
54                                 printf("%f\n", f(a, c_n));
55                                 break;
56                         default:
57                                 printf("ERROR!\n");
58                 }
59                 if ((f(a, a_n) * f(a, c_n)) <= 0)
60                         b_n = c_n;
61                 else
62                         a_n = c_n;
63         }
64         printf("\n");
65 }
66
```

```
67 int main(void)
68 {
69         double  a;      /* x = sqrt(a) の解を求める */
70         double  x_0;    /* newton 法の初期値 */
71         double  a_n;    /* 二分法の境界値 */
72         double  b_n;    /* a_n < x < b_n と設定している */
73
74         scanf("%lf", &a);
75         scanf("%lf", &x_0);
76         scanf("%lf", &a_n);
77         scanf("%lf", &b_n);
78
79         newton(0, a, x_0);      /* x_n */
80         newton(1, a, x_0);      /* f(x_n) */
81         bisection(0, a, a_n, b_n);      /* a_n */
82         bisection(1, a, a_n, b_n);      /* b_n */
83         bisection(2, a, a_n, b_n);      /* c_n */
84         bisection(3, a, a_n, b_n);      /* f(c_n) */
85
86         return 0;
87 }
```

■**問題** 4

(1) 複素数 z に対する解析関数 $f(z)=0$ を解くための Newton 法の反復式を示す

問題 3 の (1) と同様

$$\text{反復式}\begin{cases} z &= z_0 - \dfrac{f(z_0)}{f'(z_0)} \\ z_0 &= z \end{cases}$$

(2) (1) で導出した式を、x, y に関する反復式として書き直す

$z=x+iy$、$z_0=x_0+iy_0$、$f(z)=u(x,y)+iv(x,y)$ とおく。また、

$$f'(z_0)=\frac{\partial u(x_0,y_0)}{\partial x}+i\,\frac{\partial v(x_0,y_0)}{\partial x}=u_x(x_0,y_0)+iv_x(x_0,y_0) \qquad (*)$$

とする。以下、$f(z_0)=u_0+iv_0$、$f'(z_0)=u_{x,0}+iv_{x,0}$ とおくと、(1) より、

$$\begin{aligned} x+iy &= x_0+iy_0-\frac{u_0+iv_0}{u_{x,0}+iv_{x,0}}\times\frac{u_{x,0}-iv_{x,0}}{u_{x,0}-iv_{x,0}} \\ &= x_0+iy_0-\frac{(u_0\,u_{x,0}+v_0\,v_{x,0})+i(-u_0\,v_{x,0}+v_0\,u_{x,0})}{u_{x,0}^2+v_{x,0}^2} \\ &= \left(x_0-\frac{u_0\,u_{x,0}+v_0\,v_{x,0}}{u_{x,0}^2+v_{x,0}^2}\right)+i\left(y_0-\frac{-u_0\,v_{x,0}+v_0\,u_{x,0}}{u_{x,0}^2+v_{x,0}^2}\right) \end{aligned}$$

となる。両辺の実部と虚部を比較すれば、x、y の関係式が得られる。

$\therefore$ x、y の反復式は以下のようになる。

$$x\text{ の反復式}\begin{cases} x &= x_0-\dfrac{u(x_0,y_0)\,u_x(x_0,y_0)+v(x_0,y_0)\,v_x(x_0,y_0)}{u_x(x_0,y_0)^2+v_x(x_0,y_0)^2} \\ x_0 &= x \end{cases}$$

$$y\text{ の反復式}\begin{cases} y &= y_0-\dfrac{-u(x_0,y_0)\,v_x(x_0,y_0)+v(x_0,y_0)\,u_x(x_0,y_0)}{u_x(x_0,y_0)^2+v_x(x_0,y_0)^2} \\ y_0 &= y \end{cases}$$

■**参考**　$f'(z_0)$ は、以下のようにとってもよい。

$$f'(z_0)=\frac{1}{i}\,\frac{\partial u(x_0,y_0)}{\partial y}+\frac{\partial v(x_0,y_0)}{\partial y}=-i\,u_y(x_0,y_0)+v_y(x_0,y_0) \qquad (**)$$

式 $(*)$、$(**)$ は、それぞれ実軸、あるいは虚軸に平行に近づけて得られる微分係数であり、この 2 式から、$z=z_0$ における、コーシー・リーマンの関係式が得られる。