情報処理論2a

4章「二次方程式と演算精度」

中島康彦

§4. 1 今日の作業ディレクトリを作る

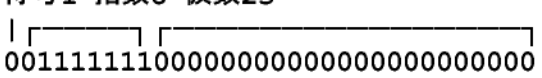
1. % cd ⇒ ホームディレクトリへ移動
2. % mkdir chap16 ⇒ ディレクトリchap16を作成
3. % cd chap16 ⇒ ディレクトリchap16へ移動
4. % netscapeを使ってdata16をchap16へダウンロード
5. % tar xvf data16 ⇒ サンプルデータの複写

```
equation1.c
equation2.c
```

§4. 2 単精度浮動小数点数(IEEE754形式の場合)

単精度浮動小数点数 … 32ビット長

符号1 指数8 仮数23



- 指数=0かつ仮数=0 … 符号付ゼロ
- 指数=0かつ仮数≠0 … 不正規化数

  (-1)^符号*2^(-126)*0.仮数

- 0<指数<255 … 正規化数

  (-1)^符号*2^(指数-127)*1.仮数

- 指数=255かつ仮数=0 … 符号付無限大(∞)
- 指数=255かつ仮数≠0 … 非数値

  例えば0/0の結果

§4. 2 単精度浮動小数点数(続き)

PI = 3.14159265358979323846

- 単精度浮動小数点数は3.141592まで正しい

```
0/1 00000000 00000000000000000000000 … +/-0
0/1 01111100 00000000000000000000000 … +/-0.125
0/1 01111101 00000000000000000000000 … +/-0.25
0/1 01111110 00000000000000000000000 … +/-0.5
0/1 01111111 00000000000000000000000 … +/-1
0/1 10000000 00000000000000000000000 … +/-2
0/1 10000000 10000000000000000000000 … +/-3
0/1 10000000 10010010000111111011010 … +/-3.14159250
0/1 10000000 10010010000111111011011 … +/-3.14159274
0/1 10000000 10010010000111111011100 … +/-3.14159297
0/1 10000000 11111111111111111111111 … +/-3.99999976
0/1 10000001 00000000000000000000000 … +/-4
0/1 10000010 00000000000000000000000 … +/-8
0/1 11111111 00000000000000000000000 … +/-∞
0/1 11111111 xxxxxxxxxxxxxxxxxxxxxxx … 非数値
```

§4. 3 倍精度浮動小数点数(IEEE754形式の場合)

倍精度浮動小数点数 … 64ビット長

```
符号1 指数11 仮数52
|┌─────────┐┌──────────────────────────────────────┐
001111111111 00000000000000000000000000000...0
```

- 指数=0かつ仮数=0 … 符号付ゼロ
- 指数=0かつ仮数≠0 … 不正規化数

  (-1)^符号*2^(-1022)*0.仮数

- 0<指数<2047 … 正規化数

  (-1)^符号*2^(指数-1023)*1.仮数

- 指数=2047かつ仮数=0 … 符号付無限大(∞)
- 指数=2047かつ仮数≠0 … 非数値

  例えば0/0の結果

§4. 3 倍精度浮動小数点数(続き)

PI = 3.14159265358979323846

- 倍精度浮動小数点数は3.141592653589793まで

```
0/1 00000000000 00000000000000000000000...0000 … +/-0
0/1 01111111100 00000000000000000000000...0000 … +/-0.125
0/1 01111111101 00000000000000000000000...0000 … +/-0.25
0/1 01111111110 00000000000000000000000...0000 … +/-0.5
0/1 01111111111 00000000000000000000000...0000 … +/-1
0/1 10000000000 00000000000000000000000...0000 … +/-2
0/1 10000000000 10000000000000000000000...0000 … +/-3
0/1 10000000000 10010010000111111011010...0111 … +/-3.14159265358979267
0/1 10000000000 10010010000111111011010...1000 … +/-3.14159265358979311
0/1 10000000000 10010010000111111011010...1001 … +/-3.14159265358979356
0/1 10000000000 11111111111111111111111...1111 … +/-3.99999999999999955
0/1 10000000001 00000000000000000000000...0000 … +/-4
0/1 10000000010 00000000000000000000000...0000 … +/-8
0/1 11111111111 00000000000000000000000...0000 … +/-∞
0/1 11111111111 xxxxxxxxxxxxxxxxxxxxxxx...xxxx … 非数値
```

- 有限桁に丸めたための誤差 ⇒ 丸め誤差

§4. 4 二次方程式の求解

$$Ax^2 + Bx + C = 0$$

機能設計(入出力形式)

標準入力	… A, B, C
標準出力	… 解(無い場合はN.A.), 左辺値

構成設計(内部データ構造)

Aを記憶する変数	… double型 a
Bを記憶する変数	… double型 b
Cを記憶する変数	… double型 c
判別式を記憶する変数	… double型 d
解を記憶する変数	… double型 x1, x2

§4. 5 二次方程式の解を計算するには

数式による表現を試みる

- $x1 = \dfrac{-B+\sqrt{(B^2 - 4AC)}}{2A}$

- $x2 = \dfrac{-B-\sqrt{(B^2 - 4AC)}}{2A}$

§4. 6 詳細設計(equation1.c)

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

main()
{
  double a, b, c, d;
  double x1, x2;

  while (scanf("%lf %lf %lf", &a, &b, &c) == 3) {
    d = b*b - 4.0*a*c;
    if (d >= 0.0) {
      x1 = (-b+sqrt(d))/(2.0*a);
      x2 = (-b-sqrt(d))/(2.0*a);
      printf("x1=%27.20e (%27.20e)\n", x1, a*x1*x1+b*x1+c);
      printf("x2=%27.20e (%27.20e)\n", x2, a*x2*x2+b*x2+c);
    }
    else {
      printf("x1=N.A.\n");
      printf("x2=N.A.\n");
    }
  }
  exit(0);
}
```

§4. 7 解説

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

main()
{
  double a, b, c, d;
  double x1, x2;
  :
  exit(0);
}
```

/usr/include/math.h

- 関数sqrt()の定義が入っている.

内部変数の宣言

- 浮動小数点数型(64ビット)

§4. 7 解説(続き)

```
while (scanf("%lf %lf %lf", &a, &b, &c) == 3) {
:
}
```

連続して解を求める.

- scanf()の返り値が3である間繰り返す.
- double型変数へ読み込むには, 書式%lfを用いる.

小数点以下がない形式	… 1
指数部がない形式	… 1.0
指数部がある形式	… 1.0e+00

§4. 7 解説(続き)

```
d = b*b - 4.0*a*c;
if (d >= 0.0) {
  x1 = (-b+sqrt(d))/(2.0*a);
  x2 = (-b-sqrt(d))/(2.0*a);
  printf("x1=%27.20e (%27.20e)\n", x1, a*x1*x1+b*x1+c);
  printf("x2=%27.20e (%27.20e)\n", x2, a*x2*x2+b*x2+c);
}
else {
  printf("x1=N.A.\n");
  printf("x2=N.A.\n");
}
```

判別式≧0. 0であれば

- x1 = (-b+sqrt(d))/(2.0*a);
- x2 = (-b-sqrt(d))/(2.0*a);

解および左辺値の表示

- %27.20e … 27桁表示(小数点以下20桁)

§4. 8 コンパイルと実行

1．コンパイルする.

```
% gcc equation1.c -o equation1 -lm
```

2．実行

```
% ./equation1
1 -2 1
x1= 1.000000000000000000e+00 ( 0.000000000000000000e+00)
x2= 1.000000000000000000e+00 ( 0.000000000000000000e+00)
1 -1000 1
x1= 9.999899999999000034222e+02 ( 1.164153218269348144 53e-10)
x2= 1.000001000022621155 95e-03 (-2.062106041478273255 05e-11)★
1 -1000000 1
x1= 9.999999999899999238 6e+05 ( 0.000000000000000000e+00)
x2= 1.000007614493370056 15e-06 (-7.614492369967251761 41e-06)★
1 -1000000000 1
x1= 1.000000000000000000e+09 ( 1.000000000000000000e+00)
x2= 0.000000000000000000e+00 ( 1.000000000000000000e+00)★
```

- ★の部分の誤差が大きい

§4. 9 桁落ち

桁落ち … 有効数字桁数が減ること

- B》4ACの場合，$|B|-\sqrt{(B^2 - 4AC)}$ の有効数字桁数が極端に減る.
- この例ではx2の精度が悪化している.
- A=1 B=-1000000 C=1の場合,

```
|-1000000.0|
      =1000000.000000000000000 (0x412e848000000000)
√(1000000000000.0 - 4.0)
      = 999999.999997999984771 (0x412e847fffffbce4)
差
      =       0.000002000015229 (0x3ec0c70000000000)
                                  ~~~~~~~~~~
```

この部分の情報が欠落する

§4. 10 二次方程式の求解(改良版)

$|B|-\sqrt{(B^2 - 4AC)}$を使わない表現

- $x1 = \dfrac{|B|+\sqrt{(B^2 - 4AC)}}{2A}$ … (B<0)

- $x1 = \dfrac{-|B|-\sqrt{(B^2 - 4AC)}}{2A}$ … (B≧0)

- $x2 = \dfrac{C}{Ax1}$

§4. 11 詳細設計(equation2.c)

```
main()
{
      :途中同じ

       x1 = (fabs(b)+sqrt(d))/(2.0*a);
       x1 = (b<0.0)?x1:-x1;
       x2 = c/(a*x1);

      :途中同じ
}
```

§4. 12 解説

```
x1 = (fabs(b)+sqrt(d))/(2.0*a);
x1 = (b<0.0)?x1:-x1;
x2 = c/(a*x1);
```

x1の計算

- 関数fabs()およびsrqt()はdouble型を引数としdouble型を返す
- fabs()の定義は/usr/include/math.hに含まれる.
- b<0.0が真である時，?演算子の値はx1，偽である時，-x1となる.

§4. 13 コンパイルと実行

1．コンパイルする.

```
% gcc equation2.c -o equation2 -lm
```

2．実行

```
% ./equation2
1 -2 1
x1= 1.00000000000000000000e+00 ( 0.00000000000000000000e+00)
x2= 1.00000000000000000000e+00 ( 0.00000000000000000000e+00)
1 -1000 1
x1= 9.99989999999000034222e+02 ( 1.16415321826934814453e-10)
x2= 1.00001000000200006431e-03 ( 0.00000000000000000000e+00)●
1 -1000000 1
x1= 9.99999999998999992386e+05 ( 0.00000000000000000000e+00)
x2= 1.00000000000100008890e-06 ( 0.00000000000000000000e+00)●
1 -1000000000 1
x1= 1.00000000000000000000e+09 ( 1.00000000000000000000e+00)
x2= 1.00000000000000006228e-09 ( 0.00000000000000000000e+00)●
```

- 桁落ちが改善されている.

§4. 14 例題

equation2.cでは，x1が0.0である場合に0除算例外が発生し，プログラムが異常終了する．x1が0.0となるのは，入力データがどのような条件を満たす場合かを考えよ．

また，0除算例外を発生することなく正しい解を表示するためには，どのように改良すれば良いかを示せ．

さらに，他にもプログラムが異常終了する可能性があれば，同様の対処を考えよ．

§4. 15 今日の課題

次の式により，aに+0.0，bに+∞が格納される．

- a = 0.0;
- b = 1.0/a;

この時，以下のc～gがどのような値となるか実験せよ．

- c = a/a;
- d = b/b;
- e = a/b;
- f = b/a;
- g = a*b;

宛先: nakashim@econ.kyoto-u.ac.jp
件名: unix2-学生番号

今日はここまで