数値計算

数値計算とは？

Mathematica を
実行してみよう

- 代数演算
  - 等式を一定の規則のもとに変形して解析的に求める
- 数値計算
  - 反復計算によって”近似的に”解を求める
  - 解析的に求めることが困難な方程式を解く場合に用いる

例） 物理学、化学、天文学などにおける数値積分や微分方程式の解法など

数値計算トピック

- 数値積分
  - モンテカルロ法を含む。
- 数値計算における誤差
- 連立方程式

数値積分

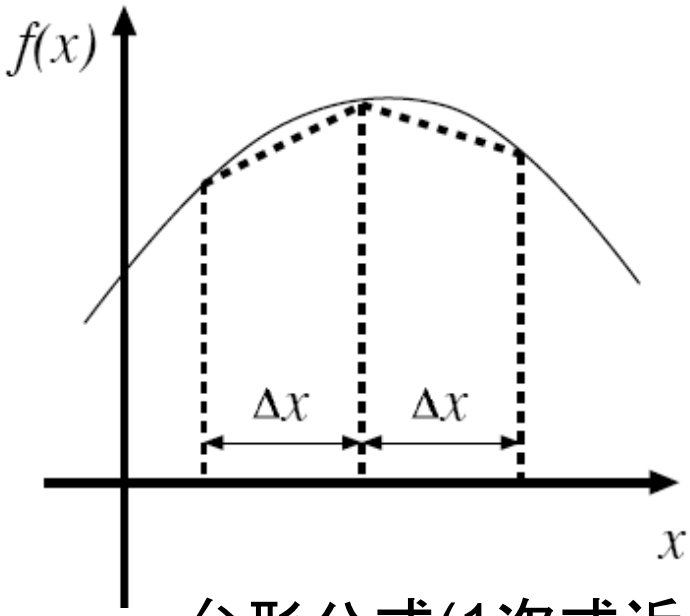
1. 台形公式
2. シンプソン公式
3. モンテカルロ法

進捗状況の確認

1. 台形公式とシンプソン公式の両方を実施した
2. 台形公式だけ実施した
3. まだ書いていない

シンプソン公式

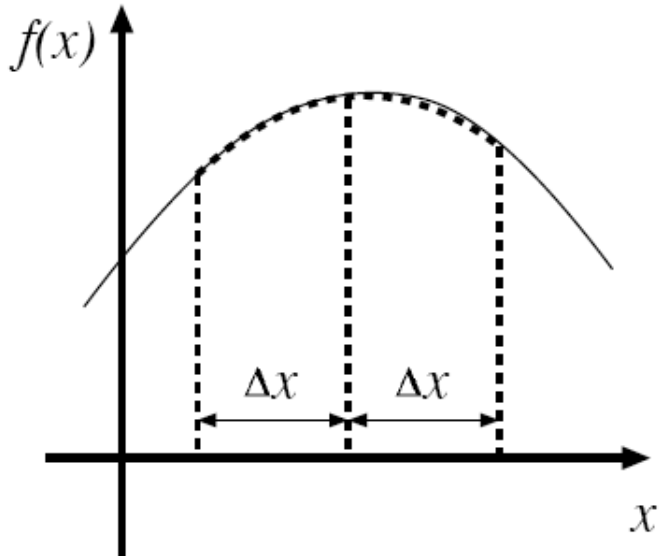
- 積分を1次式ではなく2次式で近似する方法



台形公式(1次式近似)

$$\frac{1}{2}\{f(x)+f(x+\Delta x)\}\Delta x$$

$$\Delta x = \frac{x_e - x_s}{n}$$

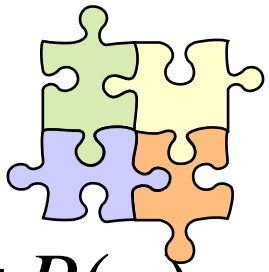


シンプソン公式(2次式近似)

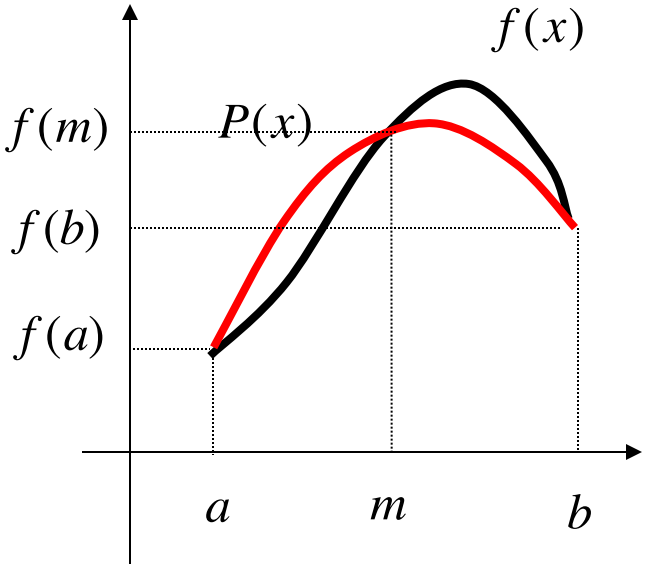
$$\frac{1}{3}\{f(x)+4f(x+\Delta x)+f(x+2\Delta x)\}\Delta x$$

$$\Delta x = \frac{x_e - x_s}{2n}$$

ラグランジュ補間



- ある関数$f(x)$を3点a,m,bで補間する2次関数$P(x)$

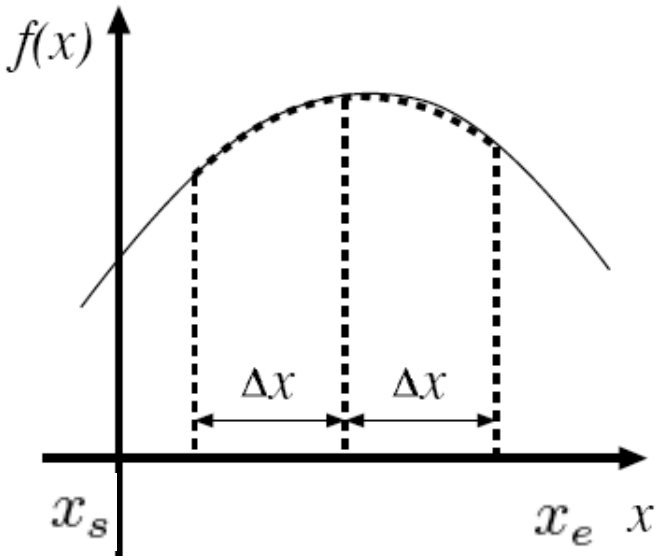


$$P(x)=\frac{(x-b)(x-m)}{(a-b)(a-m)}f(a)+\frac{(x-a)(x-b)}{(m-a)(m-b)}f(m)+\frac{(x-a)(x-m)}{(b-a)(b-m)}f(b)$$

$$m=\frac{(b+a)}{2} \Rightarrow \int_a^b P(x)dx=\frac{(b-a)}{6}[f(a)+4f(m)+f(b)]$$

$$a\to x,\quad b\to x+2\Delta x,\quad m\to x+\Delta x \Rightarrow = \frac{1}{3}\{f(x)+4f(x+\Delta x)+f(x+2\Delta x)\}\Delta x$$

シンプソン公式（続き）



部分区間 $\Delta x = \dfrac{x_e - x_s}{2n}$ とし

区間 $2\Delta x$ の部分の積分近似値は

$$\frac{1}{3}\left\{f(x) + 4f(x+\Delta x) + f(x+2\Delta x)\right\}\Delta x$$

となるため、x_s から x_e までの

積分近似値の総和は

$$\begin{aligned}\int_{x_s}^{x_e} f(x)dx &\approx \sum_{i=0}^{n-1}\frac{1}{3}\left\{f(x_s+2i\Delta x) + 4f(x_s+(2i+1)\Delta x) + f(x_s+(2i+2)\Delta x)\right\}\Delta x \\ &= \frac{\Delta x}{3}\left\{f(x_s) + 4f(x_s+\Delta x) + f(x_e) + \sum_{i=1}^{n-1}\left(2f(x_s+2i\Delta x) + 4f(x_s+(2i+1)\Delta x)\right)\right\}\end{aligned}$$

練習

- 以下の積分をシンプソン公式で近似して円周率を求めよ。n を引数として円周率を返す関数を定義せよ。

$$\pi = 4\int_0^1 \sqrt{1-x^2}\,dx$$

trapezoid.rb (simpson.rb) の f を定義しなおし、

```
def pi(n)
    4.0 * trapezoid または simpson(0.0, 1.0, n)
end
```

1. シンプソン公式で π の計算ができたら投票してください。

進捗状況の確認

1. シンプソン公式までできた。
2. 台形公式までできた。
3. プログラムは動いたが π にならない。
4. まったく動いていない

乱数とモンテカルロ法

1. モンテカルロ法
2. 乱数とは？
   - 一様乱数
   - 正規乱数
3. 疑似乱数

モンテカルロ法

- 数学的問題の解法に乱数（疑似乱数）を用いる方法の総称をモンテカルロ法という

  モンテカルロとはモナコにある賭博で有名な都市（賭博場が国営）であり、多くの賭博は乱数を利用して行われるためこのように呼ばれる

- 一般に以下の２種類の問題を扱う
  - 決定論的問題
    - 乱数を用いる多次元積分

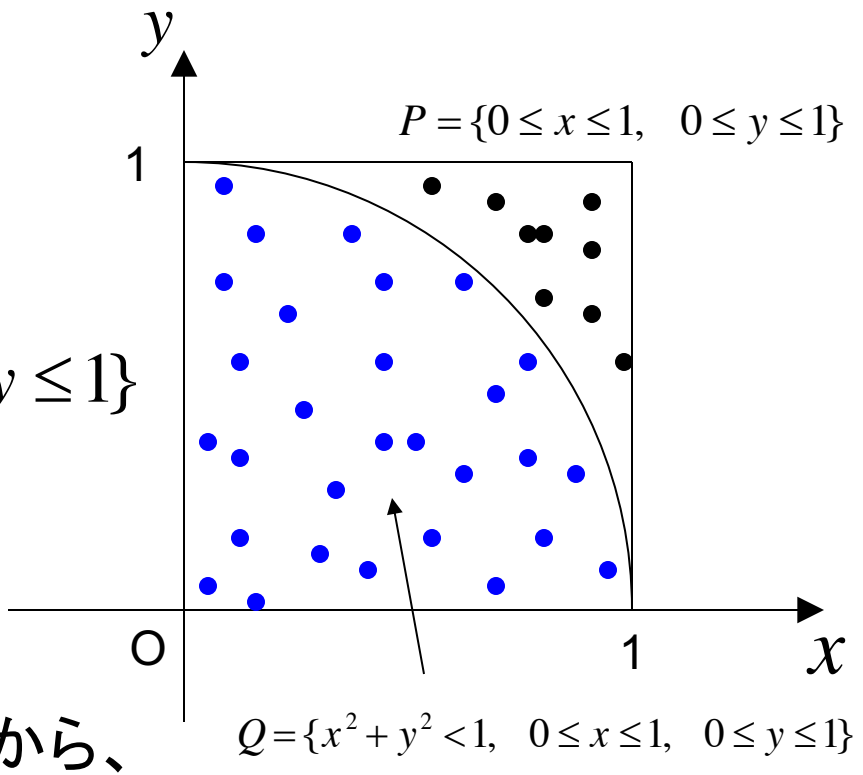
      例えば、最も簡単な例では円周率の近似計算などがある。
  - 非決定論的（確率的変動を含む）問題
    - 自然科学・社会科学におけるシミュレーション

      （実際に実験が困難であったり多大な費用がかかる場合）

      例えば、ランダムウォーク（ブラウン運動）

円周率の近似計算

- 平面上の正方領域
  $P=\{0\le x\le 1,\quad 0\le y\le 1\}$
  にランダムに n 個の点を配置し、
  その中で、四半円領域
  $Q=\{x^2+y^2<1,\quad 0\le x\le 1,\quad 0\le y\le 1\}$
  にある点の数 m を求める。
- 一様乱数を用いる場合、
  配置される点の数は
  領域の面積に比例するはずであるから、
  $m/n\approx Q/P=\pi/4$
- つまり円周率は $\pi\approx 4m/n$ で
  近似できる。



$x_i\in[0,1)\quad y_i\in[0,1)\quad i=1..n$ のうち

- $x_i^2+y_i^2\ge 1$ を満たす点
- $x_i^2+y_i^2<1$ を満たす点

ただし、精度を上げるためには、
非常に多くの点を用いる必要がある

乱数とは？

- 乱数とは、ある一つの数が現れたとき、それに続く数が前の数と全く関係なく現れるような列（乱数列）の要素

もう少し丁寧に定義すると…

- 乱数列とは、ある分布に従う（互いに独立な事象を表す）確率変数の実現値の列をいう
- 乱数とは、乱数列の各要素である

ある確率変数 x の実現値の列である乱数列

$\{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots\}$ において

x が x_{n+1} の値をとることは $x_1, x_2, \ldots, x_n$ からは予測できない

確率分布によって、一様乱数、正規乱数などがある。

一様乱数

- 出現確率が値によらず一定である乱数を一様乱数という（確率分布が一様である場合）

例えば、

サイコロの振って出た数字を並べたものは、続けて出現する二つの数の間には因果関係がないため乱数列である

また、（サイコロに細工がしてなければ）1から6までの数字が1/6の（一定の）確率で出現するため一様乱数である

年末ジャンボ宝くじの当選番号（数字部分）を毎年記録したものは一様乱数だろうか？

矢を射ることによって全く公平に行われるとするとこれも一様乱数となる。

自然現象における乱数列

放射性原子は放射線を出して崩壊する
（原子核が崩壊して別の核種になるか、励起状態の原子核
がより低いエネルギー状態へ遷移する）
放射性物質の単位時間あたりの崩壊数を計測する

- 個々の原子核は他と無関係に崩壊するとすると乱数列
- 同一の核種は平均として同じ割合で崩壊するが、
  崩壊数は同じ確率で出現するわけではない
  （平均からのずれがある）

このように、一般には、一様ではない乱数列を扱う必要もある

正規乱数

- 出現確率が正規分布に従うような乱数を正規乱数という
- 確率変数 x が ξ よりも小さい値をとる確率 $P(x)$ が以下の正規分布に従う場合

$$P(x<\xi)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\xi}\exp\left(-\frac{x^2}{2}\right)dx$$

自然現象を扱うシミュレーション（計算機で模擬実験を行うこと）では、一様乱数と共に正規乱数をよく用いる

疑似乱数
(pseudorandom numbers)

- 乱数列のように見えるが、計算機である計算を行うことによって求めることができる数列をいう
- 区間[0,1)の一様乱数を近似する数列をさすことが多い

疑似乱数は、*n*個目までの*m*個の既知要素 $x_n, x_{n-1}, \ldots, x_{n-m+1}$ を用いて以下の漸化式により*n+1*番目の要素を計算する

$$x_{n+1} = f(x_n, x_{n-1}, \ldots, x_{n-m+1})$$

疑似乱数列 $\{x_n\}$ は周期が大きいことが望ましい

（実際には、一様分布に関する適合度、統計的独立性などを様々な乱数検定法で検定し疑似乱数としての資格を与える）

```
def mcplot(n)
    a = make2d(500,500)
    for i in 1..n
        x = rand() # random number in [0,1)
        y = rand()
        if x*x + y*y < 1.0
            a[y*500][x*500] = 1.0
        else
            a[y*500][x*500] = 0.4
        end
    end
    a
end
```

```
def montecarlo(n)
    m = 0
    for i in 1..n
        x = rand() # random number in [0,1)
        y = rand()
        if x*x + y*y < 1.0
            m = m + 1
        end
    end
    m*1.0/n
end
```

montecarlo.rb

練習

- show(mcplot(n)) を実行して、点がばらまかれる様子を観察せよ。
  - 配列の引数に浮動小数点数を与えると切り捨てられる。
- 4*montecarlo(n) を色々な引数で実行し、円周率が近似できることを確かめよ。

1. できたら投票してください。

数値計算トピック

- 数値積分
  - モンテカルロ法を含む。
- 数値計算における誤差
- 連立方程式

数値計算における誤差

- 実数データ表現
  - 浮動小数点表現
- 丸め誤差
  - 0.1の2進数表記・単精度表現・倍精度表現
- 桁落ち
- 情報落ち
- 打切り誤差
  - Taylor展開・級数展開の高次項の影響

実数データ表現

- 計算機内の数値データは，有限長のビット列で表現される
- 大きな数や小さな数を表現するためには，
  符号部と指数部，仮数部をもった実数表現(浮動小数点表現)が利用される

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m(2)} \times 2^{d'_1 d'_2 \cdots d'_{c(2)} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

$s_{(2)}$　符号

$1.d_1 d_2 \cdots d_{m(2)}$　仮数

$d'_1 d'_2 \cdots d'_{c(2)} - b$　指数

この表記は、例えば、1.0101…010を2進数として解釈するという意味である

b　バイアス（指数が負の値をとれるようにするもの）

単精度と倍精度

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m_{(2)}} \times 2^{d'_1 d'_2 \cdots d'_{c_{(2)}} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

- 単精度　32ビットで実数を表現
- 倍精度　64ビットで実数を表現

	符号部	指数部	仮数部	m	c	b
単精度 (32 ビット)	第 0 ビット	第 1〜8 ビット	第 9〜31 ビット	23	8	127
倍精度 (64 ビット)	第 0 ビット	第 1〜11 ビット	第 12〜63 ビット	52	11	1023

指数部と仮数部がすべて0の実数は0（±0）を意味する

このほかにも±無限大などの表現が可能

整数の大小比較回路がそのまま使える

次の数は 10 進数で何？

0 00000000 00000000000000000000000

(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

実数データ表現

- 計算機内の数値データは，有限長のビット列で表現される
- 大きな数や小さな数を表現するためには，
  符号部と指数部，仮数部をもった実数表現(浮動小数点表現)が利用される

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m(2)} \times 2^{d'_1 d'_2 \cdots d'_{c(2)} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

$s_{(2)}$　符号

$1.d_1 d_2 \cdots d_{m(2)}$　仮数

$d'_1 d'_2 \cdots d'_{c(2)} - b$　指数

この表記は、例えば、1.0101…010を2進数として解釈するという意味である

b　バイアス（指数が負の値をとれるようにするもの）

単精度と倍精度

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m_{(2)}} \times 2^{d'_1 d'_2 \cdots d'_{c_{(2)}} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

- 単精度　32ビットで実数を表現
- 倍精度　64ビットで実数を表現

	符号部	指数部	仮数部	m	c	b
単精度 (32 ビット)	第 0 ビット	第 1〜8 ビット	第 9〜31 ビット	23	8	127
倍精度 (64 ビット)	第 0 ビット	第 1〜11 ビット	第 12〜63 ビット	52	11	1023

指数部と仮数部がすべて0の実数は0（±0）を意味する

このほかにも±無限大などの表現が可能

整数の大小比較回路がそのまま使える

127は2進数で何桁？

(単精度表現)

1. 1
2. 11
3. 111
4. 1111
5. 11111
6. 111111
7. 1111111
8. 11111111
9. 111111111

次の数は 10 進数で何？

0 01111111 00000000000000000000000
(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

次の数は 10 進数で何？

0 10000000 00000000000000000000000

(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

実数データ表現

- 計算機内の数値データは，有限長のビット列で表現される
- 大きな数や小さな数を表現するためには，
  符号部と指数部，仮数部をもった実数表現(浮動小数点表現)が利用される

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m(2)} \times 2^{d'_1 d'_2 \cdots d'_{c(2)} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

$s_{(2)}$　符号

$1.d_1 d_2 \cdots d_{m(2)}$　仮数

$d'_1 d'_2 \cdots d'_{c(2)} - b$　指数

この表記は、例えば、1.0101…010を2進数として解釈するという意味である

b　バイアス（指数が負の値をとれるようにするもの）

1/2 は 2 進数では？

1. 0.1
2. 0.01
3. 0.011111111111…
4. 0.010101010101…
5. 0.010010010010…
6. 0.001100110011…
7. 0.001001001001…

1/4 は 2 進数では？

1. 0.1
2. 0.01
3. 0.011111111111…
4. 0.010101010101…
5. 0.010010010010…
6. 0.001100110011…
7. 0.001001001001…

1/3 は 2 進数では？

1. 0.1
2. 0.01
3. 0.011111111111…
4. 0.010101010101…
5. 0.010011001100…
6. 0.001100110011…
7. 0.001001001001…

1/5 は 2 進数では？

1. 0.1
2. 0.01
3. 0.011111111111…
4. 0.010101010101…
5. 0.010010010010…
6. 0.001100110011…
7. 0.001001001001…

実数データ表現

- 計算機内の数値データは，有限長のビット列で表現される
- 大きな数や小さな数を表現するためには，
  符号部と指数部，仮数部をもった実数表現(浮動小数点表現)が利用される

IEEE 754　実数表現に関する規格

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m(2)} \times 2^{d'_1 d'_2 \cdots d'_{c(2)} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

$s_{(2)}$　符号

$1.d_1 d_2 \cdots d_{m(2)}$　仮数

$d'_1 d'_2 \cdots d'_{c(2)} - b$　指数

この表記は、例えば、1.0101…010を2進数として解釈するという意味である

b　バイアス（指数が負の値をとれるようにするもの）

次の数は 10 進数で何？

0 10000000 10000000000000000000000

(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

次の数は 10 進数で何？

0 01111110 00000000000000000000000

(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

次の数は 10 進数で何？

0 10000001 01000000000000000000000

(単精度表現)

1. 0.0
2. 0.1
3. 0.2
4. 0.3
5. 0.5
6. 1.0
7. 2.0
8. 3.0
9. 5.0

単精度表現

- 仮数部23ビット 指数部8ビット

仮数の表現範囲は有効桁数は24ビット
これは10進数で7桁程度の精度しかないことを意味する

$1.00\ldots0_{(2)} = 1$ から $1.11\ldots1_{(2)} = 2-2^{-23} \approx 2-1.2\times10^{-7} \approx 2$

指数の表現範囲は 1.2×10^{-38} から 1.7×10^{38}

$2^{00000001_{(2)}-127} = 2^{-126} \approx 1.2 \times 10^{-38}$

$2^{11111110_{(2)}-127} = 2^{127} \approx 1.7 \times 10^{38}$

$2^{00000000_{(2)}-127}$ と、
$2^{11111111_{(2)}-127}$ は
特別な意味を持つので
除外してある

最大値の限界は $\approx 2 \times 2^{127} = 2^{128} \approx 3.4 \times 10^{38}$ となる

倍精度表現

- 仮数部52ビット 指数部11ビット

仮数の表現範囲は有効桁数は53ビット
これは10進数で16桁程度の精度

$1.00\ldots0_{(2)}=1$ から $1.11\ldots1_{(2)}=2-2^{-52}\approx 2-2.2\times10^{-16}\approx 2$

指数の表現範囲は 2.2×10^{-308} から 9.0×10^{307}

$2^{000\ldots01\ (2)-1023}=2^{-1022}\approx 2.2\times10^{-308}$

$2^{111\ldots10\ (2)-1023}=2^{1023}\approx 9.0\times10^{307}$

$2^{000\ldots000\ (2)-127}$ と $2^{111\ldots111\ (2)-127}$ は
特別な意味を持つので
除外してある

最大値の限界は $2\times2^{1023}=2^{1024}\approx 1.7\times10^{308}$ となる

丸め誤差の例 (教科書p.123)

irbで以下を試してみる
(0.1*3==0.3)

これは0.1の3倍が0.3に等しいかどうかを真偽で評価するための表現で
数学的には真(true)という結果になるはずであるが、実際は偽(false)となる

$$0.1_{(10)} = 0.0001\dot{1}\dot{0}0\dot{1}\dot{1}\ldots_{(2)} = 1.\dot{1}\dot{0}0\dot{1}\dot{1}_{(2)} \times 2^{-4} = 1.\dot{1}\dot{0}0\dot{1}\dot{1}_{(2)} \times 2^{01111111011_{(2)}-1023}$$

倍精度表現では 符号部 ["0", 指数部 "01111111011", 仮数部 "10011001100…110011010"] 0捨1入

$\times 3_{(10)} (\times 11_{(2)})$ ["0", "01111111101", "001100110011…00110100"] 0捨1入

$$0.3_{(10)} = 1.\dot{0}0\dot{1}\dot{1}\ldots_{(2)} \times 2^{-2} = 1.\dot{0}0\dot{1}\dot{1}_{(2)} \times 2^{01111111101_{(2)}-1023}$$

誤差

["0", "01111111101", "0011001100…1100110011"]

(検証)irbで以下を試してみよ
0.1*3-0.3
2.0**-54

丸め誤差は $\underbrace{0.0\ldots0}1_{(2)} \times 2^{-2} = 2^{-54}$
0が52個

$0.1_{(10)} = 1.\dot{1}00\dot{1}1_{(2)} \times 2^{-4}$ ["0", "01111111011", "10011001100…110011010"]

符号部　指数部　仮数部

循環小数の打ち切りで0捨1入

$3_{(10)} = 11_{(2)} = 1.1_{(2)} \times 2^{1}$

$0.1_{(10)} \times 3_{(10)}$ はどのように計算される？

$$
\begin{array}{rl}
 & 1.10011001100\ldots110011010 \\
\times) & 1.1 \times 2^{1} \\
\hline
 & 110011001100\ldots110011010 \\
+) & 110011001100\ldots110011010 \\
\hline
 & 1001100110011\ldots0110011110 \\
 & 1.001100110011\ldots0110100 \times 2^{1}
\end{array}
$$

最初の1は仮数部には入れない倍精度浮動小数点数の定義を思いだそう

52桁（倍精度の仮数部の桁数）

桁あふれで0捨1入

$$(-1)^{s_{(2)}} 1.d_1 d_2 \cdots d_{m(2)} \times 2^{d'_1 d'_2 \cdots d'_{c(2)} - b} \quad (s, d_i, d'_i \in \{0, 1\})$$

0.1x3の計算結果の倍数度表現は以下のようになる

["0", "01111111101", "00110011001…00110100"]

倍精度表現

- 仮数部52ビット 指数部11ビット

仮数の表現範囲は有効桁数は53ビット
これは10進数で16桁程度の精度

$1.00\ldots0_{(2)}=1$ から $1.11\ldots1_{(2)} = 2-2^{-52} \approx 2-2.2\times10^{-16} \approx 2$

指数の表現範囲は 2.2×10^{-308} から 9.0×10^{307}

$2^{000\ldots01\ (2)\ -1023} = 2^{-1022} \approx 2.2 \times 10^{-308}$

$2^{111\ldots10\ (2)\ -1023} = 2^{1023} \approx 9.0 \times 10^{307}$

$2^{000\ldots000\ (2)\ -127}$ と
$2^{111\ldots111\ (2)\ -127}$ は
特別な意味を持つので
除外してある

最大値の限界は $2 \times 2^{1023}=2^{1024} \approx 1.7 \times 10^{308}$ となる

桁落ちと情報落ち

- 桁落ち誤差
  - ほぼ同じような数値の差をとると有効桁数が減少する

```
  0.124
- 0.123
-------
  0.001
```

- 情報落ち誤差
  - 大きさの異なる数値の加減算では、小さな数値は大きな数値の有効桁範囲外になり無視されてしまう

```
  0.124
+ 0.0000000123
--------------
  0.124
```

桁落ち (教科書p.125)

- 桁落ち誤差
  - ほぼ同じような数値の差をとると有効桁数が減少する

```
  0.124
- 0.123
-------
  0.001
```

x = 2 における f'(x) を求める (h = 1.0 … 10^{-15})

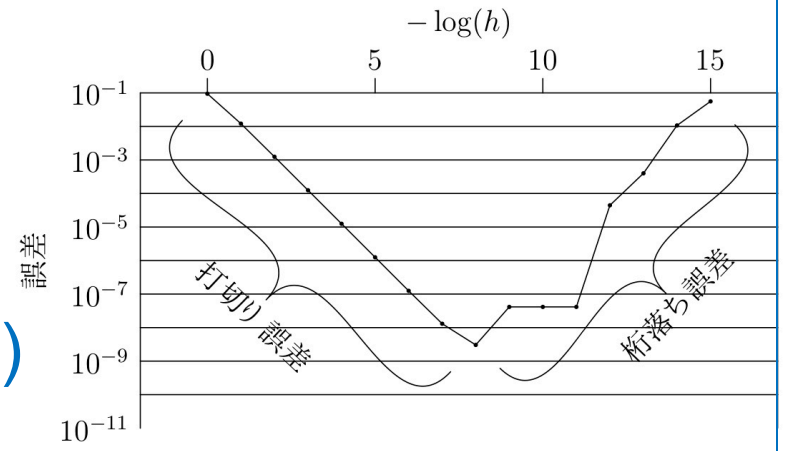
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x) = \log x \quad f'(x) = \frac{1}{x}$$

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x) = \log x \quad f'(x) = \frac{1}{x}$$

```
def f(x)
    log(x)
end
def cancellations(x,h_digits)
    errors=Array.new(h_digits+1)
    for i in 0..h_digits
        h = 0.1**i
        df=(f(x+h)-f(x))/h
        errors[i] = abs(df-1.0/x)
    end
    errors
end
```



cancellations(2.0, 15)
の結果

cancellation.rb

情報落ち (教科書p.127)

- 情報落ち誤差
  - 大きさの異なる数値の加減算では、小さな数値は大きな数値の有効桁範囲外になり無視されてしまう

```
    0.124
+ 0.0000000123
-------------
    0.124
```

数学的には恒等式だが、nが $10^6, 10^7, 10^8 \ldots$となると

$$1 = \sum_{i=1}^{n} \frac{1}{n} = \frac{n-1}{n} + \frac{1}{n}$$

単精度表現

- 仮数部23ビット 指数部8ビット

仮数の表現範囲は有効桁数は24ビット
これは10進数で7桁程度の精度しかないことを意味する

$1.00\ldots0_{(2)} = 1$ から $1.11\ldots1_{(2)} = 2-2^{-23} \approx 2-1.2\times10^{-7} \approx 2$

指数の表現範囲は 1.2×10^{-38} から 1.7×10^{38}

$2^{00000001_{(2)}-127} = 2^{-126} \approx 1.2 \times 10^{-38}$

$2^{11111110_{(2)}-127} = 2^{127} \approx 1.7 \times 10^{38}$

$2^{00000000_{(2)}-127}$ と、
$2^{11111111_{(2)}-127}$ は
特別な意味を持つので
除外してある

最大値の限界は $\approx 2 \times 2^{127} = 2^{128} \approx 3.4 \times 10^{38}$ となる

```
def coerce32(f)
  [f].pack("f").unpack("f")[0]
end
def sum(digits)
  n=10**digits
  sum = 0.0
  d = coerce32(1.0 / n)
  for i in 1..n
    sum = coerce32(sum + d)
  end
  sum
end
```

```
irb(main):004:0> sum(6)
=> 1.00903892517 09
irb(main):005:0> sum(7)
=> 1.06476747989655
irb(main):006:0> sum(8)
=> 0.25
```

lossofinformation.rb

打切り誤差 (教科書p.129)

- ある関数の値を無限級数を用いて数値計算する場合、有限の項数で打切って近似することにより生じる誤差（たとえ，実数が無限精度で表現できても生じる）

例）指数関数の原点の周りのテイラー展開

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \cdots$$

無限回の計算を行うことはできないので、有限回（n回）で打切って近似を行う

$$e^x = \sum_{i=0}^{n} \frac{x^i}{i!}$$

誤差の主要成分は $\dfrac{x^{n+1}}{(n+1)!}$