数値計算

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
- 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
- 仮数部のビット長で数値は丸められる

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）

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})

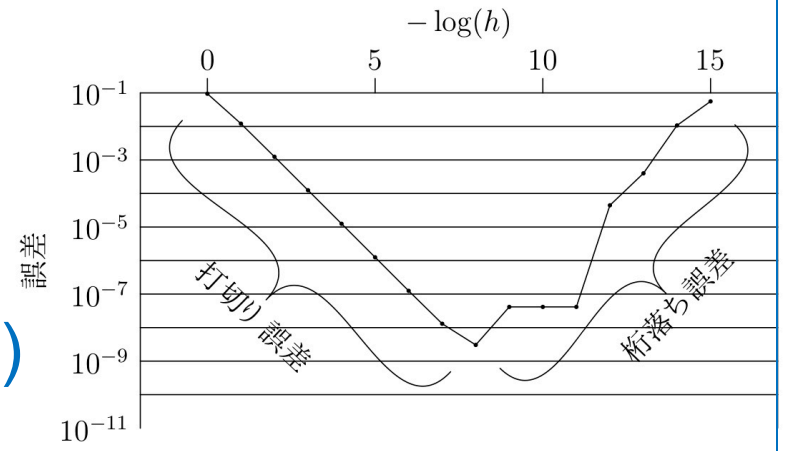
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

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

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

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

```
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              0.124
 + 0.0000000123     + 0.00123
 --------------     ---------
   0.124              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

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

```
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)
=> 1.00903892517O9
irb(main):005:0> sum(7)
=> 1.06476747989655
irb(main):006:0> sum(8)
=> 0.25
```

lossofinformation.rb

練習

桁落ち（cancellation.rb）と情報落ち（lossofinfomation.rb）のプログラムを実施して動作を確認せよ。

1. 両方できたら投票してください。

進捗状況の確認

1. 桁落ち（cancellation.rb）と情報落ち（lossofinfomation.rb）の両方を実施した
2. 桁落ち（cancellation.rb）だけ実施した
3. 情報落ち（lossofinfomation.rb）だけ実施した
4. まだ実施していない

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 $\dfrac{x^{n+1}}{(n+1)!}$

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

連立1次方程式の数値解法

- 小規模な連立1次方程式の解法
  - 消去法
    - Gauss消去法
    - Gauss-Jordan法
- （大規模な連立1次方程式の解法）
  - （反復法）
    - （Jacobi法）　講義では扱わない

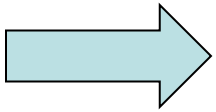
消去法による解法

- 消去法とは以下の演算を何回か行うことによって未知数を消去し、方程式をより簡単な方程式に変形して解を求める方法（筆算と同様）
  - 1つの方程式に(0でない)ある数を掛ける
  - 1つの方程式にある数を掛けて他の方程式に加える
- 多くの消去法では、未知数を1回に1個ずつ消去することによって係数行列を三角行列に変形し、後退置換によって未知数の値を逐次求める

Gauss消去法

- 与えられた連立1次方程式を行列演算で表現し、係数行列を定義する

$$\begin{aligned} x + y - z &= 2 \\ 3x + 5y - 7z &= 0 \\ 2x - 3y + z &= 5 \end{aligned}$$



$$\underbrace{\begin{pmatrix} 1 & 1 & -1 \\ 3 & 5 & -7 \\ 2 & -3 & 1 \end{pmatrix}}_{\text{係数行列}} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} 2 \\ 0 \\ 5 \end{pmatrix}}_{\text{定数ベクトル}}$$

連立1次方程式

消去法の説明を簡略化するため右のような表記を導入する

筆算で式1に3を掛けて式2から減算する操作を $r_2 - 3r_1$ で表す

1	1	−1	2	r_1
3	5	−7	0	r_2
2	−3	1	5	r_3

Gauss消去法（続き）

係数行列を三角行列に変形できるように各行に演算操作を行う

変形前

1	1	−1	2	r_1
3	5	−7	0	r_2
2	−3	1	5	r_3

例えば ○ で示された係数を0にするため

行1に3を掛けて行2から減算する $r_2 - 3r_1$

同様に ○ で示された係数を消去

するために $r_3 - 2r_1$ という操作を行う

変形後

1	1	−1	2	r_1
0	2	−4	−6	$r_2 - 3r_1$
0	−5	3	1	$r_3 - 2r_1$

この操作を左のように表記する

一番右の列は、変形における行の演算操作を表す

Gauss消去法（続き）

1	1	−1	2	r_1
0	2	−4	−6	r_2
0	−5	3	1	r_3

今度は○で示された係数を調整する

対角成分は1に、非対角成分は0になるように行の演算操作を行う

1	1	−1	2	r_1
0	1	−2	−3	$r_2/2$
0	0	−7	−14	$r_3+5/2r_2$

r_1, r_2, r_3 はその都度あらわす値が変わっていることに注意

1	1	−1	2	r_1
0	1	−2	−3	r_2
0	0	1	2	$-r_3/7$

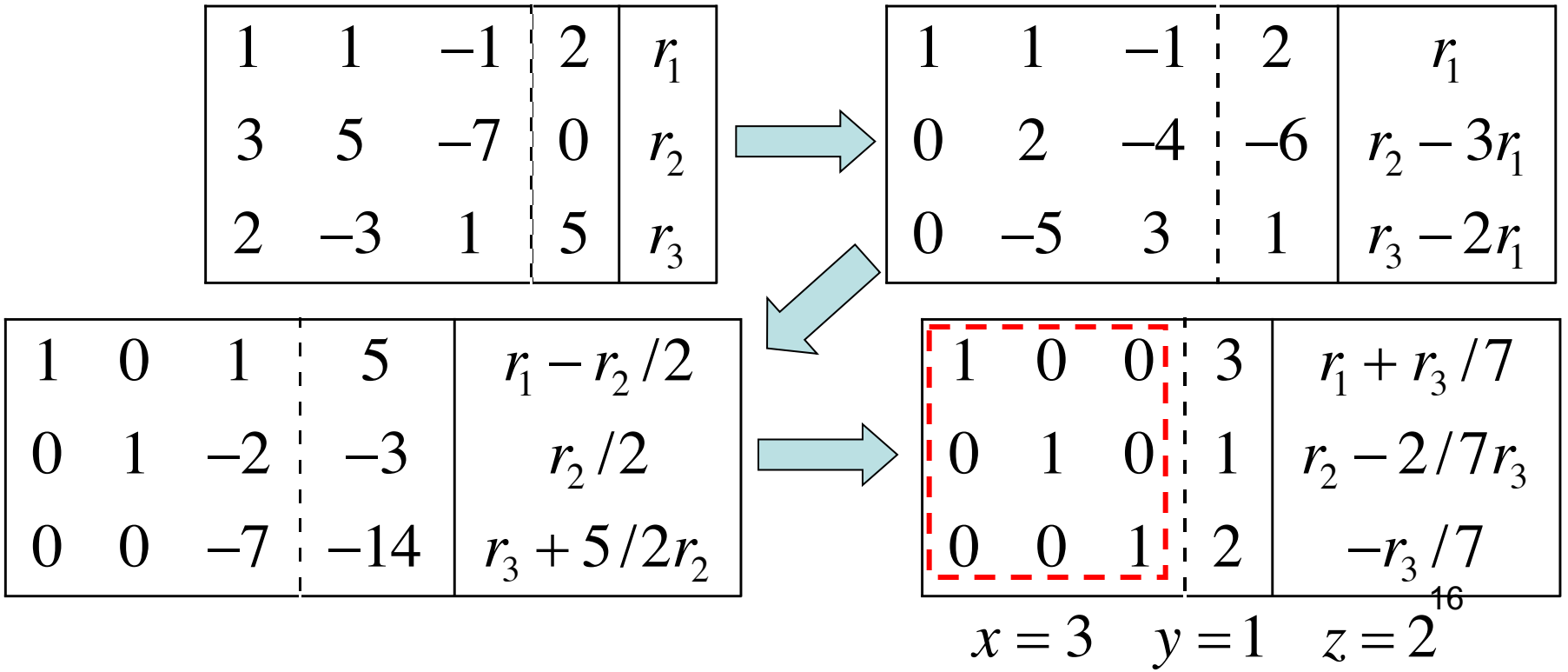
最終的に係数行列が上三角行列に変形されたら、後退置換により、各変数の値が求まる

$$x=3 \quad y=1 \quad z=2$$

Gauss-Jordan法

- Gauss消去法において、係数行列を単位行列に変形する方法
- 後退置換が不必要である

例) 前出の連立方程式の解法



```
def gj(a) # Gauss - Jordan method without pivoting
    row = a.length()
    col = a [0].length()
    for k in 0..(col-2)
        akk = a[k][k]
        for i in 0..(col-1) # normalize row k
            a[k][i]=a[k][i ]*1.0/akk
        end
        for i in 0..(row-1) # eliminate column k
            if i != k # of all rows but k
                aik = a[i][k]
                for j in k..(col-1)
                    a[i][j] = a[i][j] - aik * a[k][j]
                end
            end
        end
    end
    a
end
```

1行ずつ消去する
(k行目の処理をする)

k行目を対角成分a[k][k]で
割り算し正規化

a[i][k]を消去する処理
（ただしiはk以外）

k行にaik=a[i][k]をかけて
i行から引くことで
a[i][k]を消去（0にする）

gj.rb

Pivoting

- 消去法では、係数行列の要素での割算における問題がある
  - 割算の分母が0になる場合（係数行列の対角要素が0）
    ⇒行または変数を入れ替える必要がある
  - 割算の分母が0に近い値になる場合
    ⇒割算の結果非常に大きな数字が結果として表れると数値計算の精度が失われる（前回の講義参照）
    ⇒この場合も行または変数を入れ替える必要がある

この問題を解決するため以下のPivotingを行う

$$
\begin{array}{cc}
 & \begin{array}{ccc} & \vdots & \vdots \end{array} \\
 & \begin{array}{ccc} 1 & \vdots & \vdots \end{array} \\
k\text{ 行} & \begin{array}{ccc} 0 & a_{kk} & a_{k,k+1} \end{array} \\
 & \begin{array}{ccc} 0 & a_{k+1,k} & a_{k+1,k+1} \end{array} \\
 & \begin{array}{ccc} \vdots & \vdots & \vdots \end{array} \\
i\text{ 行} & \begin{array}{ccc} 0 & a_{i,k} & a_{i,k+1} \end{array} \\
 & \begin{array}{ccc} \vdots & \vdots & \vdots \end{array}
\end{array}
$$

k 番目の行の処理で a_{kk} が0に近い場合、$a_{ki} \leftarrow a_{ki} / a_{kk}$ (k行)の絶対値が大きくなる　その k 行に a_{ik} をかけて i 行から引く時に、絶対値の差で情報落ち誤差が生じてしまう
そこで、a_{ik} の中で絶対値が最大の a_{pk} を選び、
k 行と p 行を入れ替える
元の連立方程式では式の順序を入れ替えることに相当する

教科書の訂正

- 6.4.3 Pivoting付きGauss-Jordan法（p.135）

  二番目の黒丸

  「大きな数どうしの引き算によって情報落ち誤差が生じてしまう」

  ↓

  「誤差が生じてしまう」

  ＜理由＞

  情報落ち誤差は，絶対値が大きな数と小さな数の間で演算した場合に小さな数が無視される.

  桁落ち誤差は，引き算によって生じるが，非常に似た数値の間での引き算で有効桁数が減る.

```
def gjp(a) # Gauss - Jordan method ~~without~~ WITH pivoting
    row = a.length()
    col = a [0].length()
    for k in 0..(col-2)
        akk  = a[k][k]
        for i in 0..(col-1) # normalize row k
            a[k][i]=a[k][i ]*1.0/akk
        end
        for i in 0..(row-1) # eliminate column k
            if i != k # of all rows but k
                aik = a[i][k]
                for j in k..(col-1)
                    a[i][j] = a[i][j] - aik * a[k][j]
                end
            end
        end
    end
    a
end
```

```
max = maxrow(a,k)
    # find absolute maximal coeff .
swap(a,k,max)
    # swap rows
```

k列目の係数の絶対値が
最大となる行を
k行目以降から探す

gjp.rb

maxrow(a,k)

maxrow([[1,2,3,0],[-3,0,1,2],[2,1,0,3]],0) → 1

maxrow([[1,2,3,0],[-3,0,1,2],[2,1,0,3]],1) → 2

maxrow([[1,2,3,0],[-3,0,1,2],[2,1,0,3]],2) → 2

```
def maxrow(a,k)
    m = k
    for i in ??..??
        if abs(a[i][k]) > abs(a[m][k])
            ????
        end
    end
    m
end
```

練習

- 以下の連立1次方程式をGauss-Jordan法で解いてみよ。Pivotingするかしないかで、結果が変わるか観察せよ。

$$x - 50y - 3z = -90$$

$$-85x + 2y - 25z = -6$$

$$79x + 5y + 30z = -1$$

- 共通資料から gj.rb と gjp.rb をダウンロード。
- gjp.rb内で maxrow(a,k) を定義する。

1. gj も gjp も動いたら投票してください。

進捗状況の確認

1. gj も gjp も動いたら投票してください。
2. maxrow（および abs）と swap を定義したが gjp が動かない。
3. maxrow（abs を使う）または swap が定義できない。
4. gj は動いた。
5. gj が動かない。
6. gj への入力が作れない。