信号 ・ データ処理のための行列とベクトル

—複素数,線形代数,統計学の基礎—

第3章　行列

2020/4/24

実装： 荒井博貴　　テスト：折原　俊貴

目次

開発環境

- mac OS Mojave 10.14.6
- Visual Studio Code 1.44.1
- python 3.7.6

実装課題 1

行列の和を
求める関数を実装する

行列の和（1/3）

同じサイズの行列 ***A,B*** による ***x*** 写像を考える

$$\boldsymbol{A} = [\ \boldsymbol{a}_1, \quad \boldsymbol{a}_2, \quad \dots \quad \boldsymbol{a}_N]$$

$$\boldsymbol{B} = \left[\ \boldsymbol{b}_1, \quad \boldsymbol{b}_2, \quad \dots \quad \boldsymbol{b}_N\right]$$

Ax+Bx はベクトルの基本演算より

$$\begin{aligned} \boldsymbol{Ax} + \boldsymbol{Bx} &= (x_1 a_1 + x_2 a_2 + \dots + x_n a_n) + (x_1 b_1 + x_2 b_2 + \dots + x_n b_n) \\ &= x_1(a_1 + b_1) + x_2(a_2 + b_2) + \dots + x_N(a_N + b_N) \end{aligned}$$

$\boldsymbol{a}_1+\boldsymbol{b}_1, \dots, \boldsymbol{a}_N+\boldsymbol{b}_N$ の

線型結合なので

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} = \left[\ \boldsymbol{a}_1 + \boldsymbol{b}_1, \quad \boldsymbol{a}_2 + \boldsymbol{b}_2, \quad \dots \quad \boldsymbol{a}_N + \boldsymbol{b}_N\right]$$

行列の和（2/3）

例1　2×2 の行列の和

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 6 & -1 \\ -3 & 5 \end{bmatrix}$$

$$\boldsymbol{A} + \boldsymbol{B} = \begin{bmatrix} 1+6 & 2+(-1) \\ 5+(-3) & 4+5 \end{bmatrix} = \begin{bmatrix} 7 & 1 \\ 2 & 9 \end{bmatrix}$$

行列の和は各成分同士の足し算

行列の和（3/3）

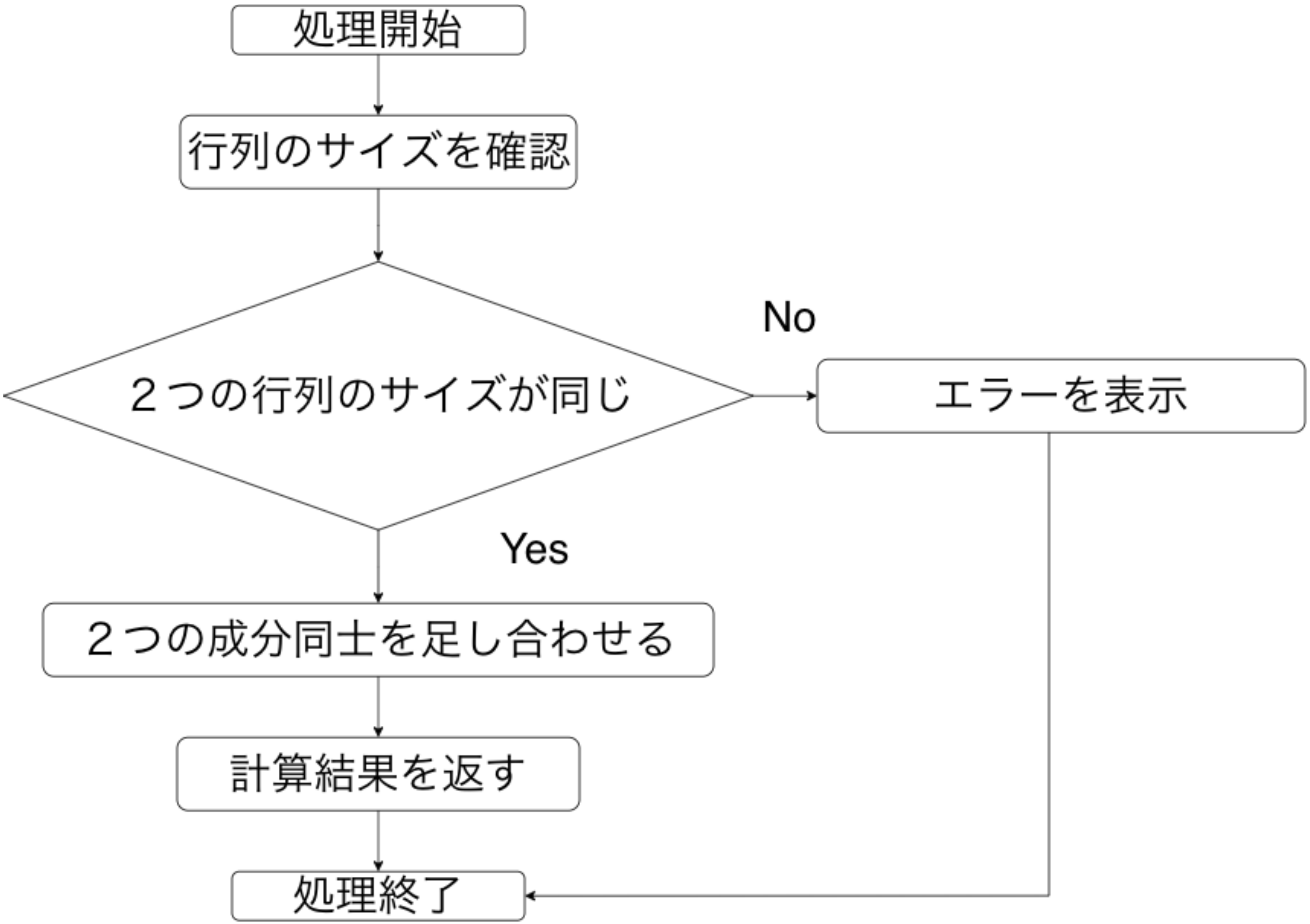
例2　2×2 の行列の差

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 6 & -1 \\ -3 & 5 \end{bmatrix}$$

$$\boldsymbol{A} - \boldsymbol{2B} = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} - 2\begin{bmatrix} 6 & -1 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} -11 & 4 \\ 11 & -6 \end{bmatrix}$$

行列の差は各成分同士の引き算

行列の和アルゴリズム（１／４）



行列の和アルゴリズム（２／４）

プログラムによる行列の定義

例　2×２ の行列

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix}$$

リストの ２ 重構造を用いて
記述すると

A=[[1,2],[5,4]]

行列の和アルゴリズム（３／４）

入力された配列のサイズを確認する方法

list の len() を用いて,リストの大きさを確認した

例

行列

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 6 \end{bmatrix}$$

リスト

A=[[1,2,3],[5,4,6]]

行： $len(\boldsymbol{A}) = 2$

列： $len(\mathbf{A[0]}) = 3$

行列の和アルゴリズム（４／４）

各成分の足し算方法

for 文を 2 重に回すことにより実現

```
for i in range(A_column):
        for j in range(A_row):
            result[i][j] = (A[i][j]+B[i][j])
```

結果
全ての場合で想定した出力と一致した

実装課題２

行列の転置とエルミート転置を
求める関数を実装する

行列の転置

行列の行と列を入れ替える作業を**転置**と呼ぶ

例

$$A = \begin{bmatrix} 1 & 2 & 7 \\ 5 & 4 & -2 \end{bmatrix}$$ **2×3**

↓ 転置

$$A^T = \begin{bmatrix} 1 & 5 \\ 2 & 4 \\ 7 & -2 \end{bmatrix}$$ **3×2**

A の転置は A^T と表す

行列のエルミート転置

複素数の行列に対して転置と複素共役を取ることを
エルミート転置と呼ぶ

例

$$A = \begin{bmatrix} 1-i & 2 & 7+i5 \\ 5 & 4+i2 & -2 \end{bmatrix}$$

2×3

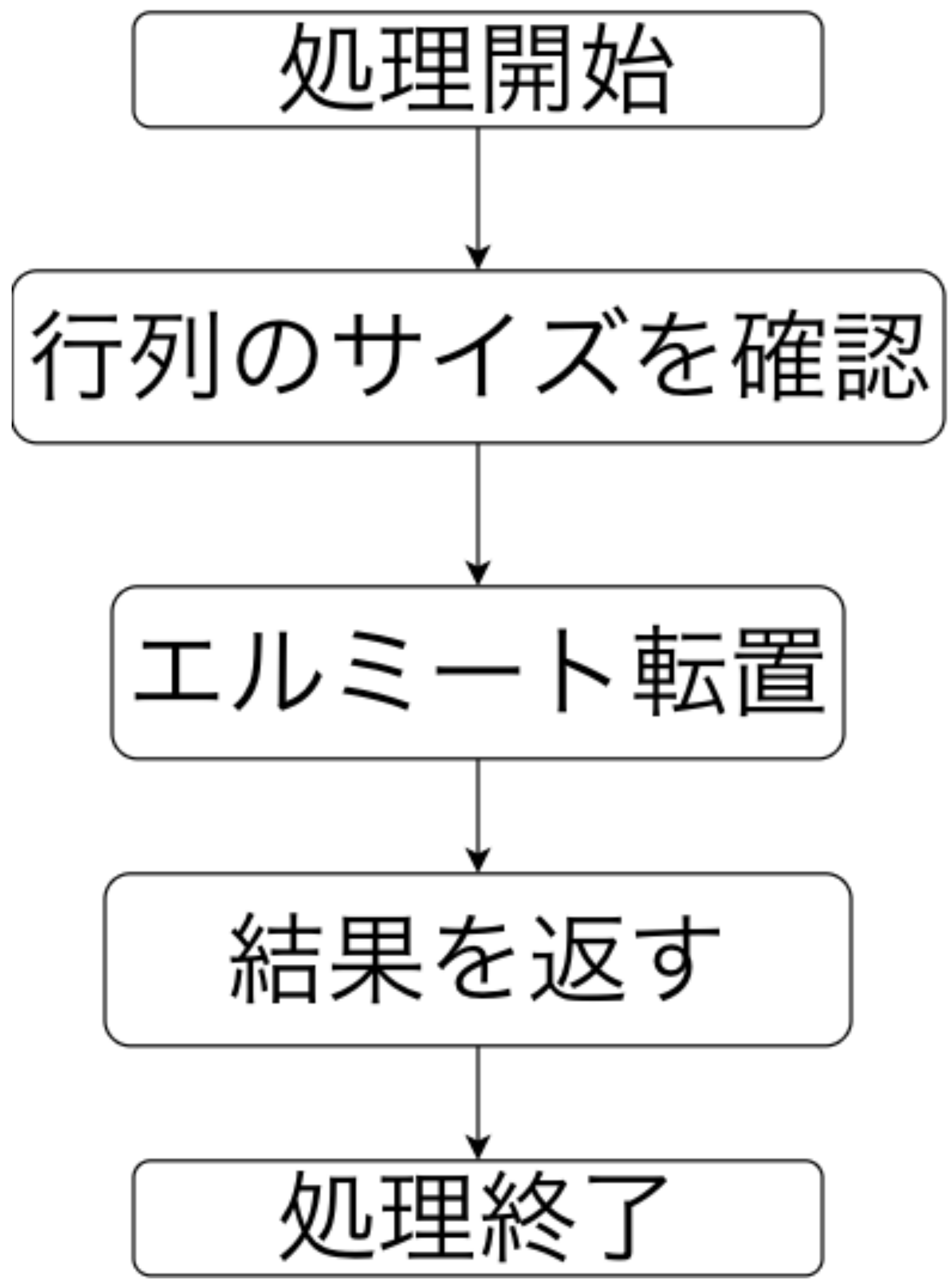
↓ 転置

$$A^H = \begin{bmatrix} 1+i & 5 \\ 2 & 4-i2 \\ 7-i5 & -2 \end{bmatrix}$$

3×2

A のエルミート転置は A^H と表す

行列の転置アルゴリズム（１／２）



行列の転置アルゴリズム（2／２）

行列の転置方法

for文を２重に回すことにより実現

```
for i in range(A_row):
        tmp = []
        for j in range(A_column):
            tmp.append(A[j][i].conjugate())
        result.append(tmp)
```

※1 **append は，リストの要素を追加していくメソッドである**

※2 **conjugate は，複素共役をとるメソッドである**

結果

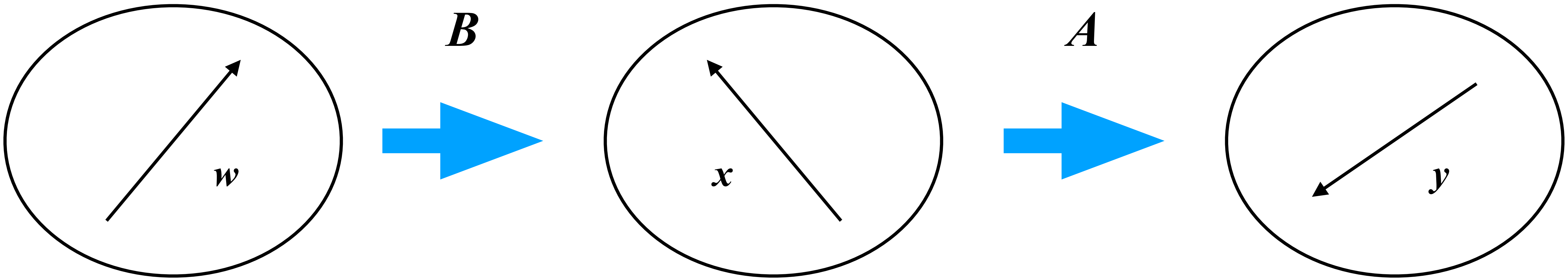
全ての場合で想定した出力と一致した

実装課題 3

行列の積を
求める関数を実装する

行列の積（１／５）

ベクトルを 2 回写像する場合を考える



$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{w} \qquad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

まとめると

$$\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{w})$$

行列の積（２／５）

行列をそれぞれ成分と列ベクトルで表すと

$$\boldsymbol{A} = (a_{mn}) = [\boldsymbol{a}_1, \quad \boldsymbol{a}_2, \quad \dots \quad \boldsymbol{a}_N]$$

$$\boldsymbol{B} = (b_{mn}) = \left[\boldsymbol{b}_1, \quad \boldsymbol{b}_2, \quad \dots \quad \boldsymbol{b}_N\right]$$

それぞれ写像を考える

$$\boldsymbol{Ax} = x_1 a_1 + x_2 a_2 + \dots + x_N a_N$$
$$\boldsymbol{Bw} = w_1 b_1 + w_2 b_2 + \dots + w_K b_K$$

$\boldsymbol{x}$ の第 n 成分は

$$x_n = w_1 b_{n1} + w_2 b_{n2} + \dots + w_K b_{nK}$$

行列の積（３／５）

$\boldsymbol{y = A(Bw)}$ に前ページの式を代入

$$\begin{aligned}\boldsymbol{y} &= (w_1 b_{11} + w_2 b_{12} + \ldots + w_k b_{1K})\boldsymbol{a}_1 \\ &\quad +(w_1 b_{21} + w_2 b_{22} + \ldots + w_k b_{2K})\boldsymbol{a}_2 + \ldots \\ &\quad +(w_1 b_{N1} + w_2 b_{N2} + \ldots + w_k b_{NK})\boldsymbol{a}_N \\ &= w_1(b_{11}\boldsymbol{a}_1 + b_{21}\boldsymbol{a}_2 + \ldots + b_{N1}\boldsymbol{a}_N) \\ &\quad +w_2(b_{12}\boldsymbol{a}_1 + b_{22}\boldsymbol{a}_2 + \ldots + b_{N2}\boldsymbol{a}_N) + \ldots \\ &\quad +w_K(b_{1K}\boldsymbol{a}_1 + b_{2K}\boldsymbol{a}_2 + \ldots + b_{NK}\boldsymbol{a}_N)\end{aligned}$$

ここで以下のように定義する

$$\boldsymbol{c}_K = b_{1K}\boldsymbol{a}_1 + b_{2K}\boldsymbol{a}_2 + \ldots + b_{NK}\boldsymbol{a}_N$$

$$\boldsymbol{y} = w_1\boldsymbol{c}_1 + w_2\boldsymbol{c}_2 + \ldots + w_K\boldsymbol{c}_K$$

行列の積（４／５）

$\boldsymbol{c}_k$ を列に持つ m×n の行列$\boldsymbol{C}$とすると

$$\boldsymbol{C} = (c_{mk}) = [\ \boldsymbol{c}_1, \quad \boldsymbol{c}_2, \quad \cdots \quad \boldsymbol{c}_K]$$

$$\boldsymbol{y} = (\boldsymbol{A}\boldsymbol{B})\boldsymbol{w} = \boldsymbol{C}\boldsymbol{w}$$

$$c_{mk} = a_{m1}b_{1k} + a_{m2}b_{2k} + \ldots + a_{mN}b_{Nk} = \sum_{n=1}^{N} a_{mn}b_{nk}$$

行列の積（５／５）

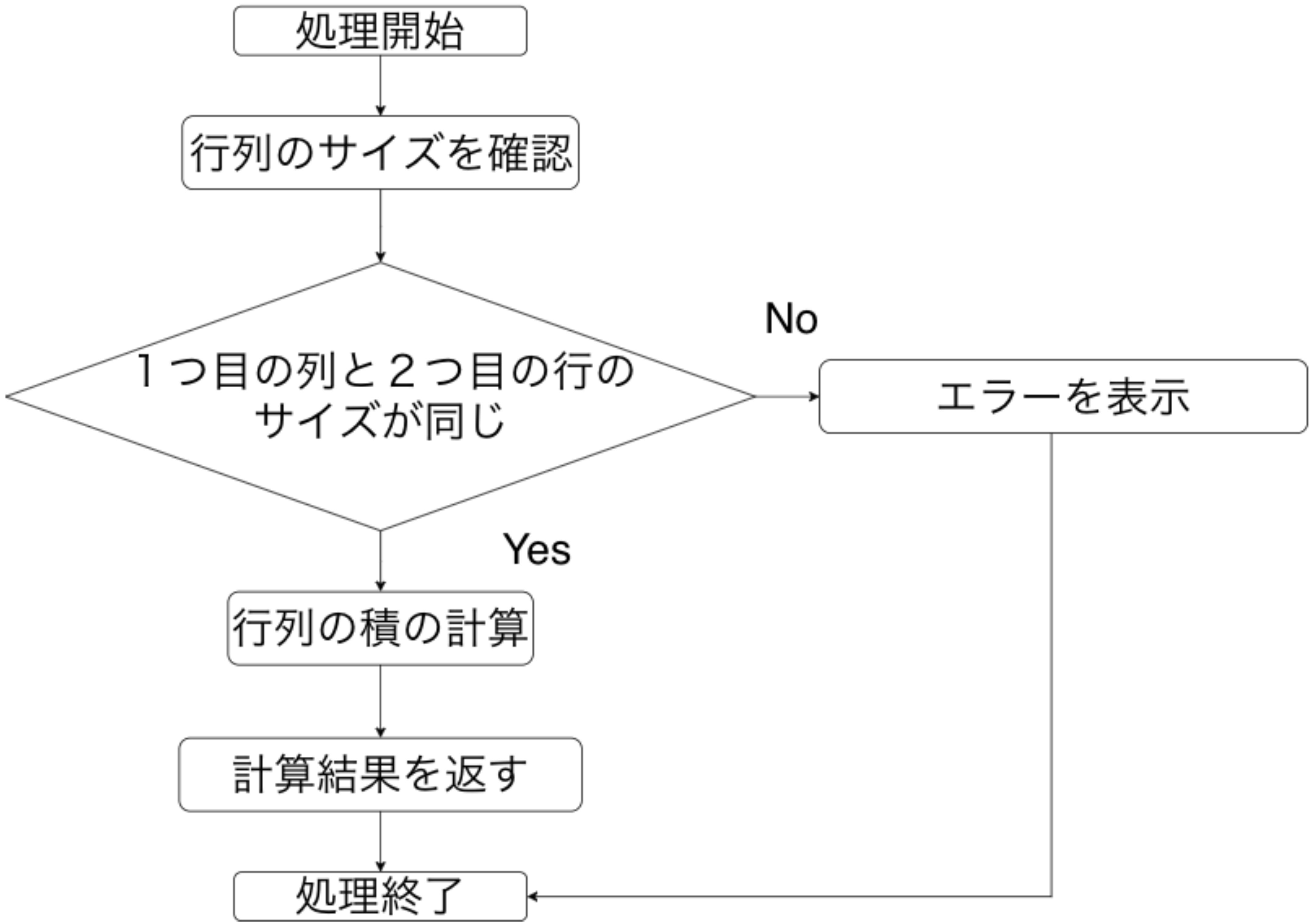
例　2×2 の行列の積

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 6 & -1 \\ -3 & 5 \end{bmatrix}$$

$$\boldsymbol{AB} = \begin{bmatrix} (1 \times 6) + (2 \times (-3)) & (1 \times (-1)) + (2 \times 5) \\ (5 \times 6) + (4 \times (-3)) & (5 \times (-1)) + (4 \times 5) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 9 \\ 18 & 15 \end{bmatrix}$$

行列の積アルゴリズム（１／２）



行列の積アルゴリズム（２／２）

行列の積計算方法

for文を３重に回すことにより実現

```
for i in range(A_column):
        for j in range(B_row):
            for k in range(A_row):
                result[i][j] += A[i][k]*B[k][j]
```

$C = AB$ のとき

行列 C の（i，j）成分は，行列 A の i 行ベクトルと

行列 B の j 列ベクトルの積で表される

結果

全ての場合で想定した出力と一致した

実装課題 4

行列が正則かどうか
判断する関数を実装する

正則の判定

ガウスの消去法

拡大行列を用いて変数を消去して，順繰りに変数を求めていく手法

手順

(1) 第 1 式を使って第 2 式以降の第 1 変数（x）を消去する

(2) 第 2 式を使って第 3 式以降の第 2 変数（y）を消去する

(3) (1)(2) を繰り返す

ガウスの消去法

例

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 0 \\ 1 & -1 & 3 & 12 \\ -1 & 5 & 2 & -1 \end{array}\right] \xrightarrow[(3) \to (3)\times 2 + (1)]{(2) \to (2)\times 2 - (1)} \left[\begin{array}{ccc|c} 2 & 1 & -1 & 0 \\ 0 & 3 & -7 & -24 \\ 0 & 11 & 3 & -2 \end{array}\right]$$

$$\xrightarrow{(3) \to (2)\times 11 - (3)\times 3} \left[\begin{array}{ccc|c} 2 & 1 & -1 & 0 \\ 0 & 3 & -7 & -24 \\ 0 & 0 & -86 & -258 \end{array}\right]$$

$$\begin{cases} 2x+y-z = 0 \\ 3y-7z = -24 \\ -86z = -258 \end{cases}$$

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

階段上の拡大行列の形を**行階段形**

ランク（階数）

ランク（階数） … 行のうち全てが 0 ではない行の数

先ほどの例では…

$$rank(\boldsymbol{A}) = 3$$

M×N の行列 $\boldsymbol{A}$ について

・M<N（横長）かつ $rank(\boldsymbol{A}) = M$ のとき，$\boldsymbol{A}$ は行についてフルランク

・M>N（縦長）かつ $rank(\boldsymbol{A}) = N$ のとき，$\boldsymbol{A}$ は列についてフルランク

・M=N（正方）かつ $rank(\boldsymbol{A}) = N$ のとき，$\boldsymbol{A}$ はフルランク

を持つという

正方行列 $\boldsymbol{A}$ がフルランクを持つとき，$\boldsymbol{A}$ を**正則行列**と呼ぶ

不定

例

$$\begin{cases} x - y + 2z = 2 \\ x + y + z = 1 \\ 2x + 2y + 2z = 2 \end{cases} \qquad \left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{array}\right]$$

(2) → (2)- (1)
(3) → (3) + (1)×2
➡

$$\left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 0 & -2 & 1 & 1 \\ 0 & -2 & 1 & 1 \end{array}\right]$$

(3) → (2)- (3)
➡

$$\left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 0 & -2 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$x = \frac{-3z+3}{2} \qquad y = \frac{z-1}{2}$$

$$\mathrm{rank}(\boldsymbol{A}) = 2,\ \mathrm{rank}(\boldsymbol{A}\,|\,\boldsymbol{f}) = 2$$

３×３ の行列において，

$rank(\boldsymbol{A}) = rank(\boldsymbol{A}\,|\,\boldsymbol{f}) = 2$ **では解が一意に定まらない**

不能

例

$$\begin{cases} x - y + 2z = 2 \\ x + y + z = 1 \\ 3x + y + 4z = 3 \end{cases} \quad \left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 3 & 1 & 4 & 3 \end{array}\right]$$

(2) → (2)- (1)

(3) → (3) + (1)×3

➡

$$\left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 0 & -2 & 1 & 1 \\ 0 & -4 & 2 & 3 \end{array}\right]$$

(3) → (3)-(2)×2

➡

$$\left[\begin{array}{ccc|c} 1 & -1 & 2 & 2 \\ 0 & -2 & 1 & 1 \\ 0 & 0 & 0 & -1 \end{array}\right]$$

連立方程式が

成り立っていない！

$$\mathrm{rank}(\boldsymbol{A}) = 2, \ \mathrm{rank}(\boldsymbol{A} \mid \boldsymbol{f}) = 3$$

３×３ の行列において,

$rank(\boldsymbol{A}) \neq rank(\boldsymbol{A} \mid \boldsymbol{f})$ **では解が存在しない**

不定と不能

$$\boldsymbol{Ax} = \boldsymbol{f}$$ (*A* は M×N の行列)

・$rank(\boldsymbol{A}) = rank(\boldsymbol{A}\,|\,\boldsymbol{f})$ のとき，解は存在し

（１）$rank(\boldsymbol{A}) = N$であれば，解は一意に決まる

（２）$rank(\boldsymbol{A}) < N$であれば，一意に決まらない（不定）

・$rank(\boldsymbol{A}) \neq rank(\boldsymbol{A}\,|\,\boldsymbol{f})$ のとき，解は存在しない（不能）

正則判定方法

例　3×3 の場合

(2) → (2)×2 - (1)
(3) → (3)×2 + (1)

$$A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & 3 \\ -1 & 5 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 1 & -1 \\ 0 & 3 & -7 \\ 0 & 11 & 3 \end{bmatrix}$$

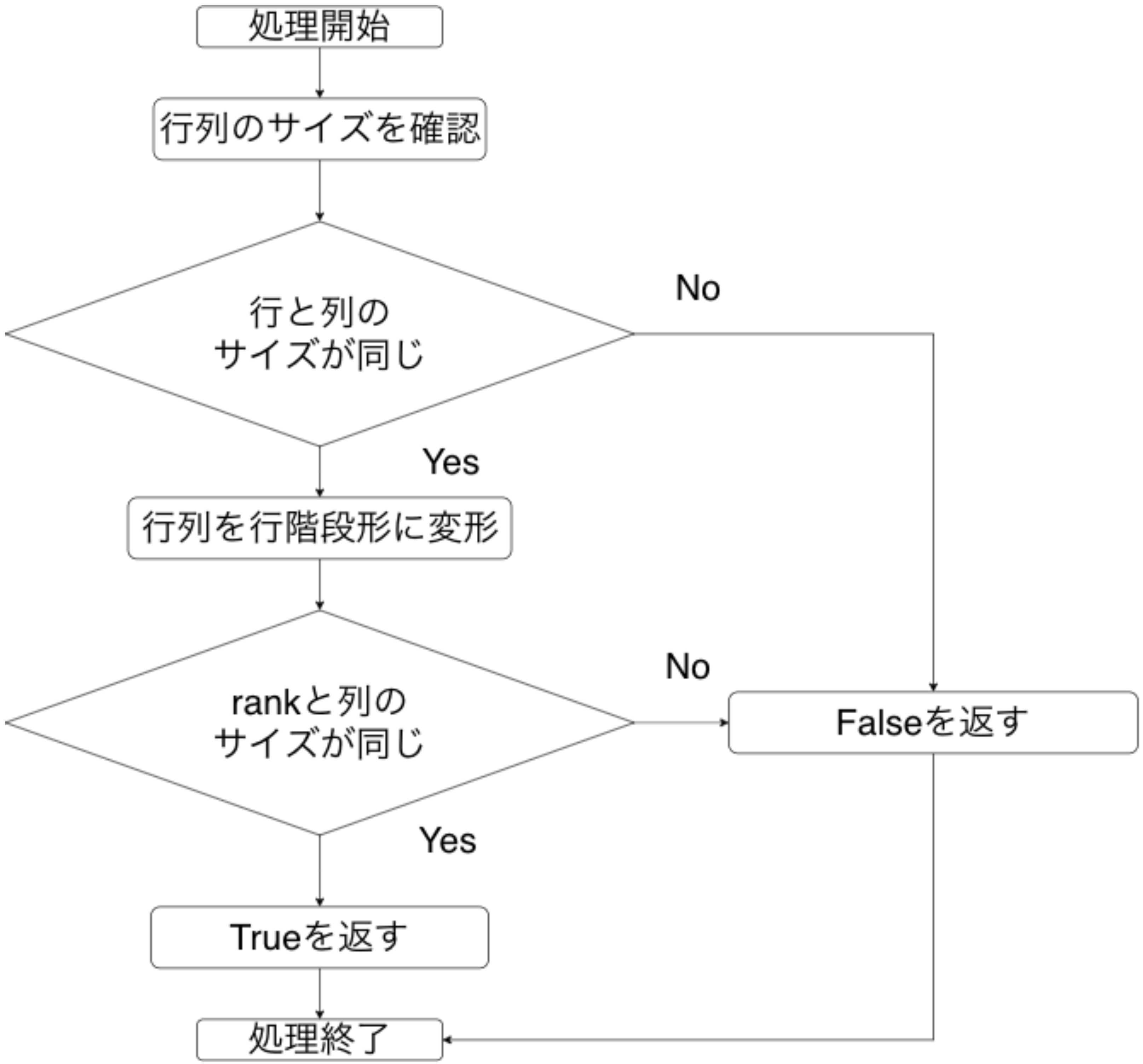
(3) → (2)×11 - (3)×3

$$\Rightarrow \begin{bmatrix} 2 & 1 & -1 \\ 0 & 3 & -7 \\ 0 & 0 & -86 \end{bmatrix}$$

$rank(\boldsymbol{A}) = 3$であるため，$\boldsymbol{A}$ は正則である

N×Nの行列 $\boldsymbol{A}$ において，$rank(\boldsymbol{A}) = N$ ならば $\boldsymbol{A}$ は**正則行列**である

正則判定アルゴリズム（１／３）



正則判定アルゴリズム（２／３）

行列を行階段形に変形する方法

for 文を用いてガウスの消去法を実現

(1) 入力された行列の 1 行目を使い，2 行目以降の第 1 成分を消去する
(2) 行列の 2 行目を使い，3 行目以降の第 2 成分を消去する
(3) 以上を繰り返す

N 行目の第 N 成分（行列の対角成分）が 0 の場合，
行を入れ替えるものを組み込んだ

正則判定アルゴリズム（３／３）

行の入れ替える方法

if 文により N 行 N 列の 0 判定を行い,

for 文を用いて N 列の N 行目以降の 0 でない行の入れ替えを実装した

```
for l in range(k+1,A_column):
    if tmp[l][k]!=0:
        for n in range(A_column):
            tmp_tmp = tmp[l][n]
            tmp[l][n] = tmp[k][n]
            tmp[k][n] = tmp_tmp
```

結果

全ての場合で想定した出力と一致した

実装課題５

連立方程式を解く関数
逆行列を求める関数
を実装する

逆行列の定義（１／２）

連立方程式は，二つの行列 $\boldsymbol{A}$，$\boldsymbol{B}$ の積 $\boldsymbol{BA}$ が単位行列 $\boldsymbol{I}$ になるような行列 $\boldsymbol{B}$ を両辺の左から乗じる操作をすることで解くことができる

$$\boldsymbol{BAx} = \boldsymbol{Bf}$$

$\boldsymbol{BA}=\boldsymbol{I}$より

$$\boldsymbol{x} = \boldsymbol{Bf}$$

このとき，$\boldsymbol{B}$ は $\boldsymbol{A}$ の逆行列であるといい，

$\boldsymbol{A}^{-1}$ と表す

逆行列の定義（２／２）

可逆

N×N の正方行列 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ があり

$$\boldsymbol{BA} = \boldsymbol{AC} = \boldsymbol{I}$$

を満たす時，$\boldsymbol{A}$ は可逆であるという（$\boldsymbol{B} = \boldsymbol{C}$）

↓ 可逆な行列 $\boldsymbol{A}$ に対して

$$\boldsymbol{AA}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$$

が成り立つ

逆行列の存在（可逆性）は，
連立方程式の解が一意に存在することを意味する

逆行列の計算法

- ガウスの消去法（掃き出し法）
- 余因子を用いた方法

→行列のサイズが大きくなると余因子を用いた方法では計算量が膨大になってしまうため実装ではガウスの消去法を用いた

ガウスの消去法

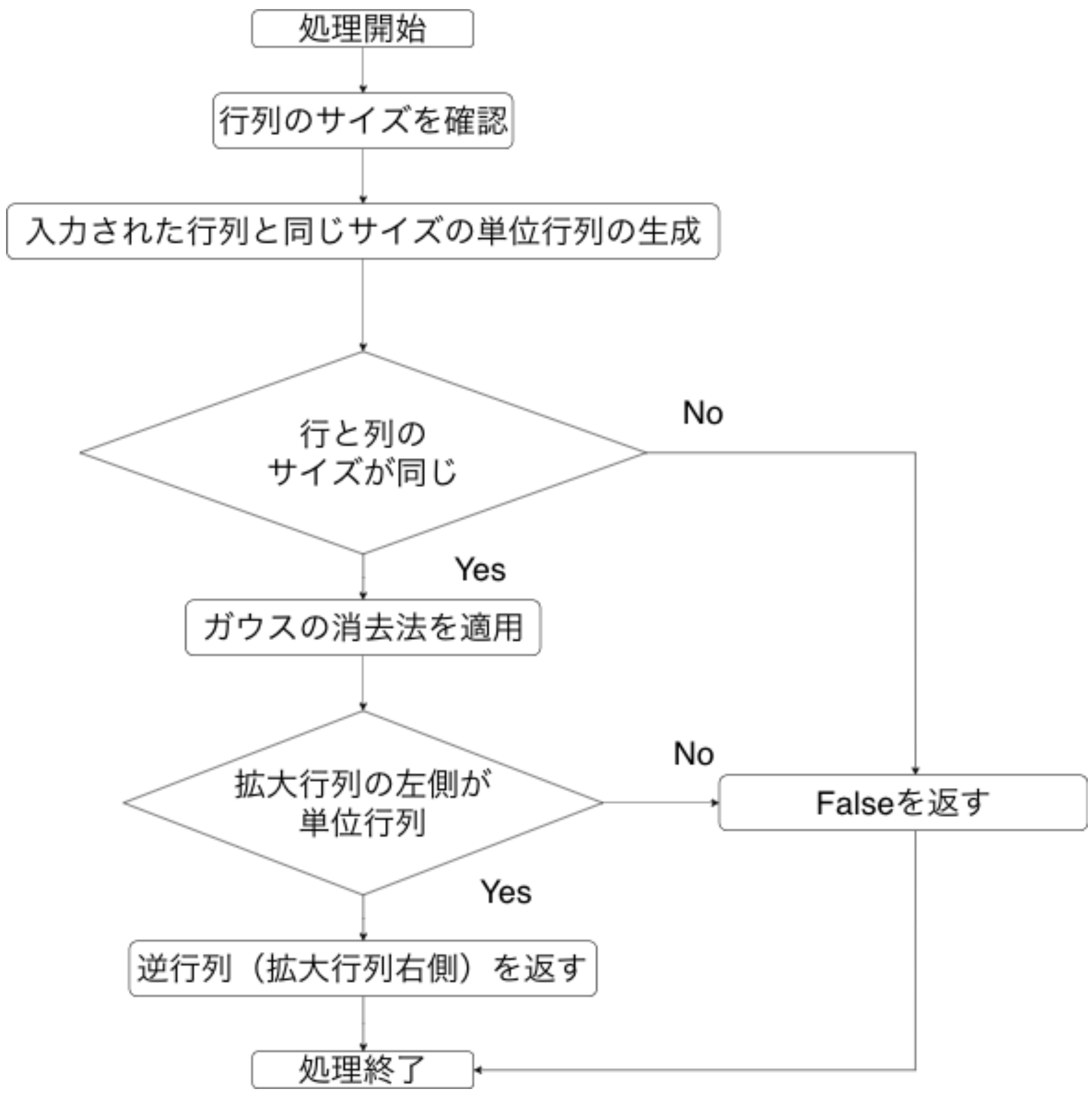
$$[\boldsymbol{A} \mid \boldsymbol{I}] = \left[\begin{array}{cc|cc} 1 & -1 & 1 & 0 \\ 2 & 5 & 0 & 1 \end{array}\right] \xrightarrow{(2)\to(2)-(1)\times 2} \left[\begin{array}{cc|cc} 1 & -1 & 1 & 0 \\ 0 & 7 & -2 & 1 \end{array}\right]$$

$$\xrightarrow[(2)\to(2)/7]{(1)\to(1)+(2)/7} \left[\begin{array}{cc|cc} 1 & 0 & \frac{5}{7} & \frac{1}{7} \\ 0 & 1 & -\frac{2}{7} & \frac{1}{7} \end{array}\right]$$

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \frac{5}{7} & \frac{1}{7} \\ -\frac{2}{7} & \frac{1}{7} \end{bmatrix}$$

行の足し引きで
逆行列を求めることができる!

逆行列計算アルゴリズム（１／２）



逆行列計算アルゴリズム（２／２）

拡大行列の変形方法

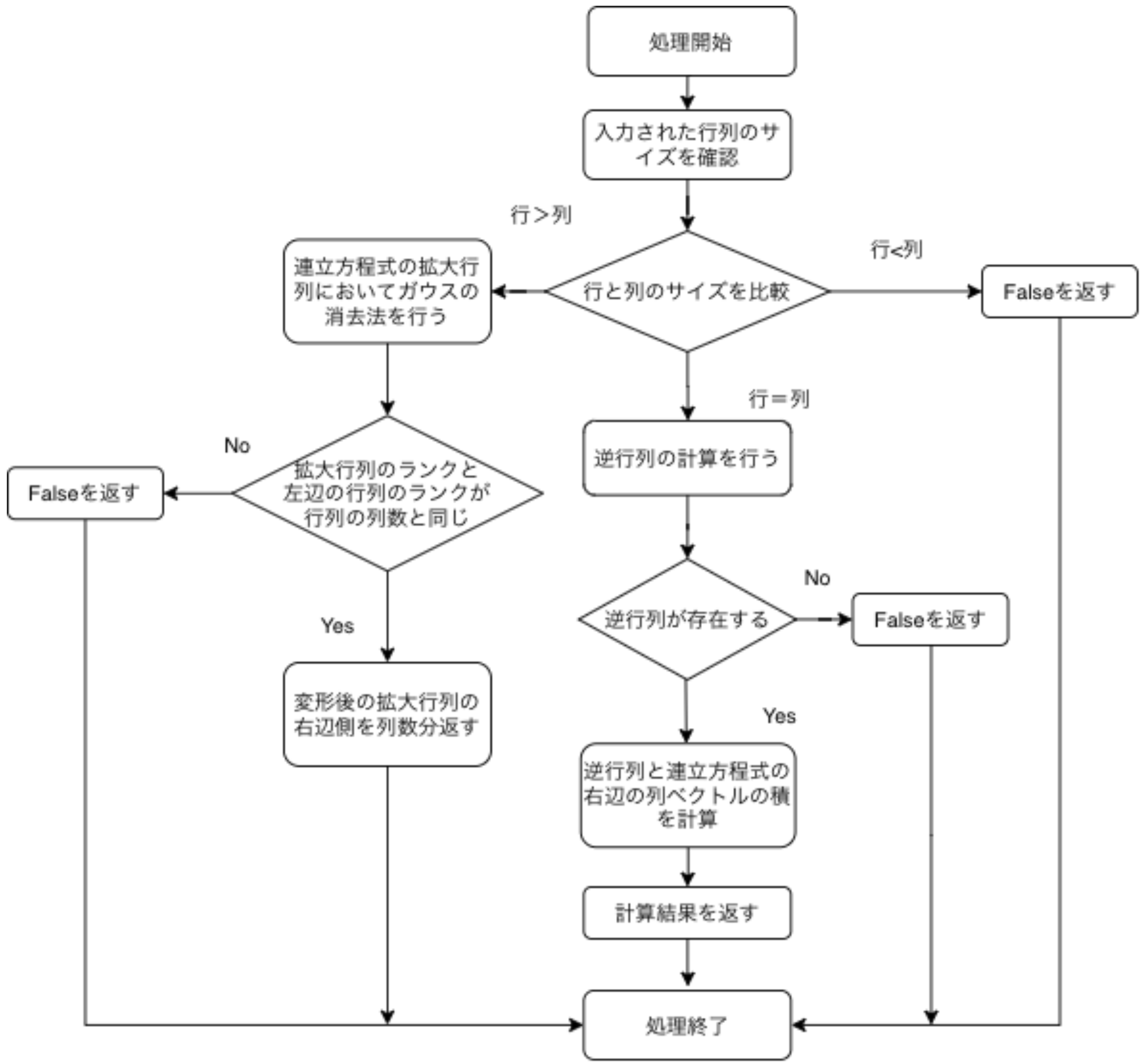
正則判定のアルゴリズムと同様

for 文を用いてガウスの消去法を実現

逆行列が存在しない場合のみ
False が返される

結果
全ての場合で想定した出力と一致した

解を求めるアルゴリズム（１／４）



解を求めるアルゴリズム（２／４）

解の計算方法（正方行列の場合）

課題３の行列の積と課題５の逆行列を計算する関数を用いて解の計算を実現した

$$\boldsymbol{Ax = f}$$

$$\boldsymbol{A^{-1}Ax = A^{-1}f}$$

$$\boldsymbol{Ix = A^{-1}f}$$

$$\boldsymbol{x = A^{-1}f}$$

解を求めるアルゴリズム（３／４）

解の計算方法（縦長の行列（行数＞列数）の場合）

ガウスの消去法を用いて
連立方程式の拡大行列を変形

ランクと列の数を比較することで
解が一意に決まるか判定した

解を求めるアルゴリズム（４／４）

例　3×2の場合

$$[\boldsymbol{A}\,|\,\boldsymbol{f}] = \left[\begin{array}{cc|c} 1 & 1 & 1 \\ 1 & 3 & 3 \\ 2 & 4 & 4 \end{array}\right]$$

(2)→(2)-(1)
(3)→(3)-2×(1)
➡

$$\left[\begin{array}{cc|c} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{array}\right]$$

(1)→(1)-(2)/2
(3)→(3)-(2)
➡

$$\left[\begin{array}{cc|c} 1 & 0 & -1 \\ 0 & 2 & 2 \\ 0 & 0 & 0 \end{array}\right]$$

$rank(\boldsymbol{A}) = rank(\boldsymbol{A}\,|\,\boldsymbol{f}) = 2$ **より，列数と一致するので解が一意に決まる**

結果
全ての場合で想定した出力と一致した

まとめ

- python の基本的な書き方（for 文など）を理解した
- 信号処理で用いられる行列の基本を学ぶことができた
- テストファーストプログラミングによる実装の仕方を学ぶことができた