第3回　連立一次方程式(1)
逆行列 ー ガウス-ジョルダン法

連立一次方程式

$$
\begin{cases}
a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\
a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\
\qquad\qquad \vdots \\
a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n
\end{cases}
$$

$$
\begin{bmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & & \\
\vdots & & \ddots & \\
a_{n,1} & & & a_{n,n}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}
$$

既知行列 → ← 既知ベクトル

↑ 未知ベクトル

$$A\,\boldsymbol{x} = \boldsymbol{b}$$

解析的に解けるが n が大きいと時間がかかる

基本行演算と基本行列

- 基本行演算 と 基本行列
  - ある行をスカラー倍（0倍は除く）する

$$i)\begin{pmatrix} a_{i1} & \cdots & a_{in} \end{pmatrix} \Rightarrow i)\begin{pmatrix} ca_{i1} & \cdots & ca_{in} \end{pmatrix}$$

A

  - この演算は次のような行列を掛けることで実現できる

$$i)\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c & \\ & & & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix}\begin{pmatrix} a_{i1} & \cdots & a_{in} \end{pmatrix} = \begin{pmatrix} ca_{i1} & \cdots & ca_{in} \end{pmatrix}$$

$E_i(c)$ A

単位行列の第 i 行を c 倍した行列

基本行演算と基本行列

- 行の交換

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{i1} & \cdots & a_{in} \\ a_{k1} & \cdots & a_{kn} \end{pmatrix} \Rightarrow \begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{k1} & \cdots & a_{kn} \\ a_{i1} & \cdots & a_{in} \end{pmatrix}$$

A

- この演算は次のような行列を掛けることで実現できる

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & & 1 \\ & & & \ddots & \\ & & 1 & & 0 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix}\begin{pmatrix} a_{i1} & \cdots & a_{in} \\ a_{k1} & \cdots & a_{kn} \end{pmatrix} = \begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{k1} & \cdots & a_{kn} \\ a_{i1} & \cdots & a_{in} \end{pmatrix}$$

$E_{i,k}$ A

単位行列の第 i 行と第 k 行を交換した行列

基本行演算と基本行列

- ある行に別の行のスカラー倍を加える

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{i1} & \dots & a_{in} \\ a_{k1} & \dots & a_{kn} \end{pmatrix} \Rightarrow \begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{i1}+ca_{k1} & \dots & a_{in}+ca_{kn} \\ a_{k1} & \dots & a_{kn} \end{pmatrix}$$

A

- この演算は次のような行列を掛けることで実現できる

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} \mathbf{1} & & & & \\ & \ddots & & & \\ & & \mathbf{1} & \mathbf{c} & \\ & & \mathbf{0} & \ddots & \\ & & & & \mathbf{1} \end{pmatrix}\begin{pmatrix} a_{i1} & \dots & a_{in} \\ a_{k1} & \dots & a_{kn} \end{pmatrix} = \begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} a_{i1}+ca_{k1} & \dots & a_{in}+ca_{kn} \\ a_{k1} & \dots & a_{kn} \end{pmatrix}$$

$E_{i,k}(c)$ A

単位行列の第 i 行に第 k 行の c 倍を加えた行列

基本行演算と基本行列

- まとめ
  - 行列 A に次の基本行演算を行うことは
    - ある行をスカラー倍（0倍は除く）する
    - 行の交換
    - ある行に別の行のスカラー倍を加える

行列 A に次の基本行列を左から掛けることで実現できる.

$$i)\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c & \\ & & & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \qquad E_i(c)$$

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & & 1 \\ & & & \ddots & \\ & & 1 & & 0 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \qquad E_{i,k}$$

$$\begin{matrix} i) \\ k) \end{matrix}\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & c \\ & & & \ddots & \\ & & 0 & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \qquad E_{i,k}(c)$$

2. 逆行列の導出

- 行列 A の逆行列 A^{-1} を計算する．A は正則（$|A| \neq 0$）とする．
- すでに分かっていること

  逆行列は基本行列の積で表現できる

$$A^{-1} = E_k E_{k-1} \cdots E_2 E_1$$

- 逆行列計算の基本方針

$$A^{-1}A = E_k E_{k-1} \cdots E_2 E_1 A = I$$

I を目指して
A に基本行演算を
行っていく

$$A \Rightarrow E_1 A \Rightarrow E_2 E_1 A \Rightarrow E_3 E_2 E_1 A \Rightarrow \vdots \Rightarrow E_k E_{k-1} \cdots E_2 E_1 A = I$$

$$I \Rightarrow E_1 I \Rightarrow E_2 E_1 I \Rightarrow E_3 E_2 E_1 I \Rightarrow \vdots \Rightarrow E_k E_{k-1} \cdots E_2 E_1 I = A^{-1}$$

同じ基本行演算を
I に対して
行っていく

3. 逆行列を求めるアルゴリズム

- 例

$$\left(\begin{array}{ccc} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 7 \end{array}\right) \quad A$$

$$\left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right) \quad I$$

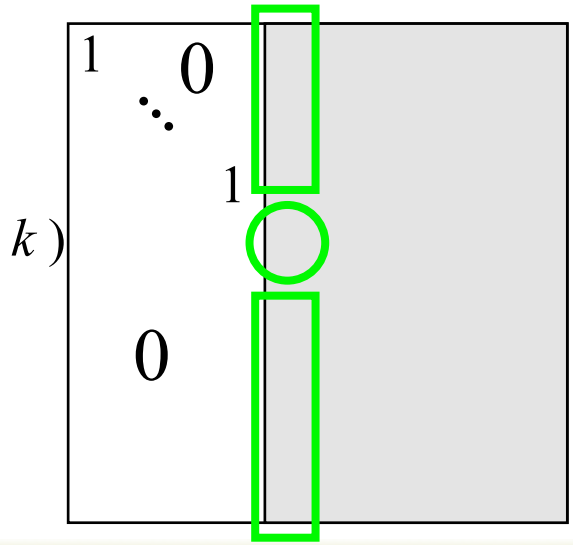
$$\left(\begin{array}{ccc} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{array}\right) \quad \left(\begin{array}{ccc} -3 & 2 & 0 \\ 2 & -1 & 0 \\ 0.5 & -1 & 0.5 \end{array}\right)$$

$$\left(\begin{array}{ccc} 1 & 2 & 3 \\ 0 & -1 & -2 \\ 0 & -2 & -2 \end{array}\right) \quad \left(\begin{array}{ccc} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{array}\right)$$

$$\left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right) \quad \left(\begin{array}{ccc} -2.5 & 1 & 0.5 \\ 1 & 1 & -1 \\ 0.5 & -1 & 0.5 \end{array}\right)$$

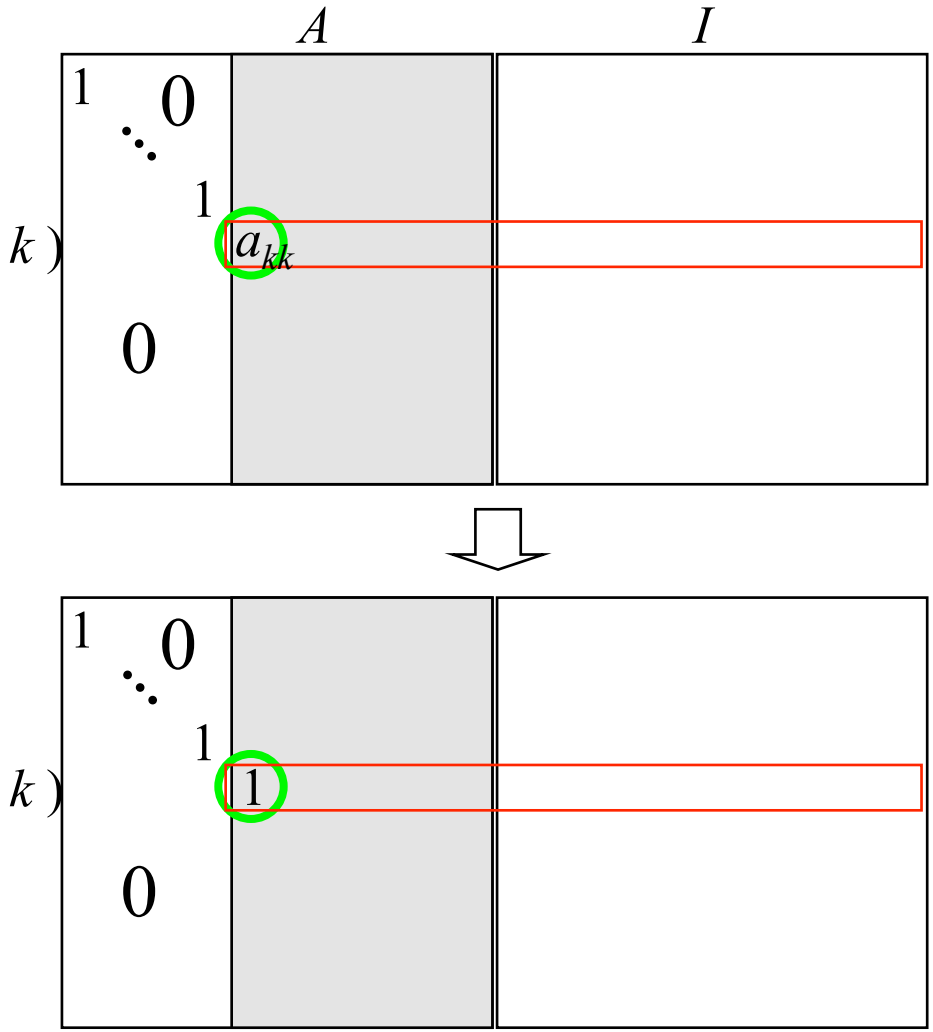
$$\left(\begin{array}{ccc} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & -2 & -2 \end{array}\right) \quad \left(\begin{array}{ccc} 1 & 0 & 0 \\ 2 & -1 & 0 \\ -3 & 0 & 1 \end{array}\right)$$

$$\left(\begin{array}{ccc} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{array}\right) \quad \left(\begin{array}{ccc} -3 & 2 & 0 \\ 2 & -1 & 0 \\ 1 & -2 & 1 \end{array}\right)$$



3. 逆行列を求めるアルゴリズム

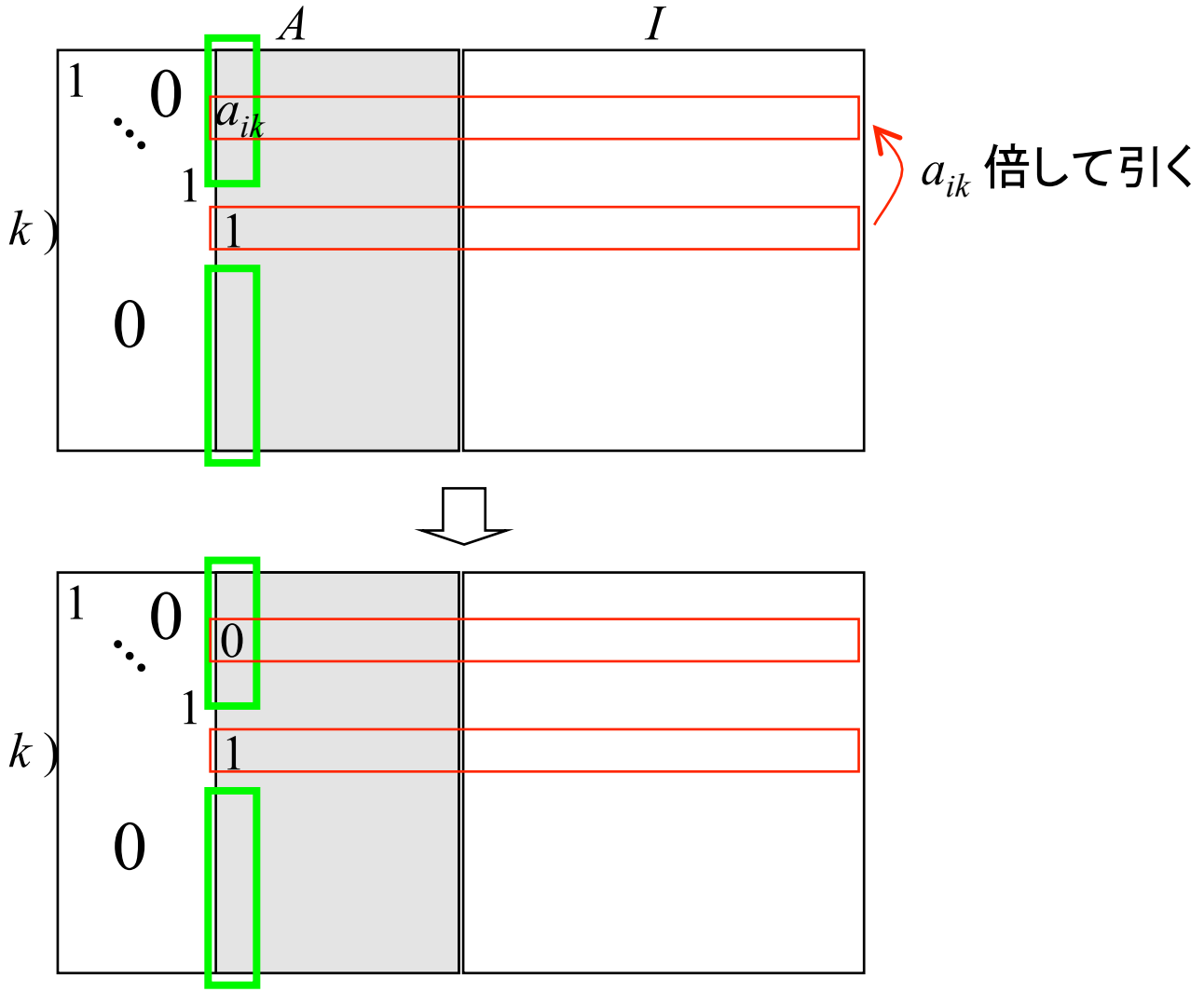
対角要素を1にする



a_{kk} で割る

3. 逆行列を求めるアルゴリズム

非対角要素を，すべて 0 にする.



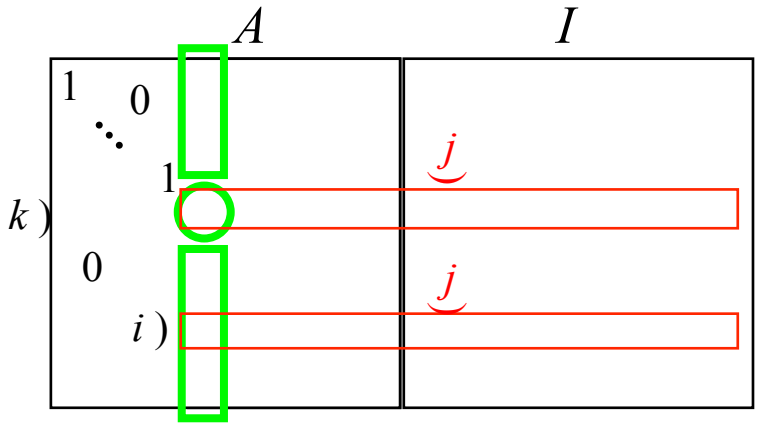
3. 逆行列を求めるアルゴリズム

注意：数学では行列の行や列の番号は「1」から始まるが，C言語の配列では「0」から始まる

C言語プログラム

```
m = 2 * n;
for (k=0; k<n; k++){
  for (j=k+1; j<m; j++)
    AI[k][j] /= AI[k][k];
  AI[k][k]=1.0;
  for (i=0; i<n ; i++){
    if (i==k) continue;
    for (j=k+1; j<m; j++)
      AI[i][j] -=
        AI[i][k] * AI[k][j];
    AI[i][k] = 0.0;
  }
}
```

[A | I] をひとつの配列で表し，その要素を
AI[k][j] (k =0,...,n–1, j =0,...,m–1)
とする



対角要素を1にする
（基本行演算「行のスカラー倍」を用いる）

非対角要素を 0 にする.
（基本行演算「ある行のスカラー倍を別の行に加える」を用いる. ）

3. 逆行列を求めるアルゴリズム

- 演習
  - 次の行列の逆行列を求めよ

$$A=\begin{pmatrix}2&3&4\\1&2&3\\3&4&7\end{pmatrix}\quad A^{-1}=\begin{pmatrix}1.0&-2.5&0.5\\1.0&1.0&-1.0\\-1.0&0.5&0.5\end{pmatrix}$$

$$A=\begin{pmatrix}2&2&8\\1&2&3\\2&3&5\end{pmatrix}\quad A^{-1}=\begin{pmatrix}-0.25&-3.5&2.5\\-0.25&1.5&-0.5\\0.25&0.5&-0.5\end{pmatrix}$$

$$A=\begin{pmatrix}8&2&2\\3&1&2\\5&2&3\end{pmatrix}\quad A^{-1}=\begin{pmatrix}0.25&0.5&-0.5\\-0.25&-3.5&2.5\\-0.25&1.5&-0.5\end{pmatrix}$$

次の節の
「ピボット選択法」で
対応が可能になる

$$A=\begin{pmatrix}1&2&3\\2&4&4\\3&4&7\end{pmatrix}$$

正則でない.
逆行列が存在しない.

$$A=\begin{pmatrix}1&2&3\\2&4&4\\3&4&7\end{pmatrix}$$

4. ガウス-ジョルダン法（掃出し法による連立一次方程式の解法）

- 連立一次方程式 $A\boldsymbol{x} = b$ を解く.

$$A\,\boldsymbol{x} = b$$

$$E_1 A\,\boldsymbol{x} = E_1 b$$

$$E_2E_1 A\,\boldsymbol{x} = E_2E_1 b$$

$$E_3E_2E_1 A\,\boldsymbol{x} = E_3E_2E_1 b$$

$$\vdots$$

$$\underline{E_kE_{k-1}\cdots E_2E_1 A}\,\boldsymbol{x} = E_kE_{k-1}\cdots E_2E_1\ b$$

$$I$$

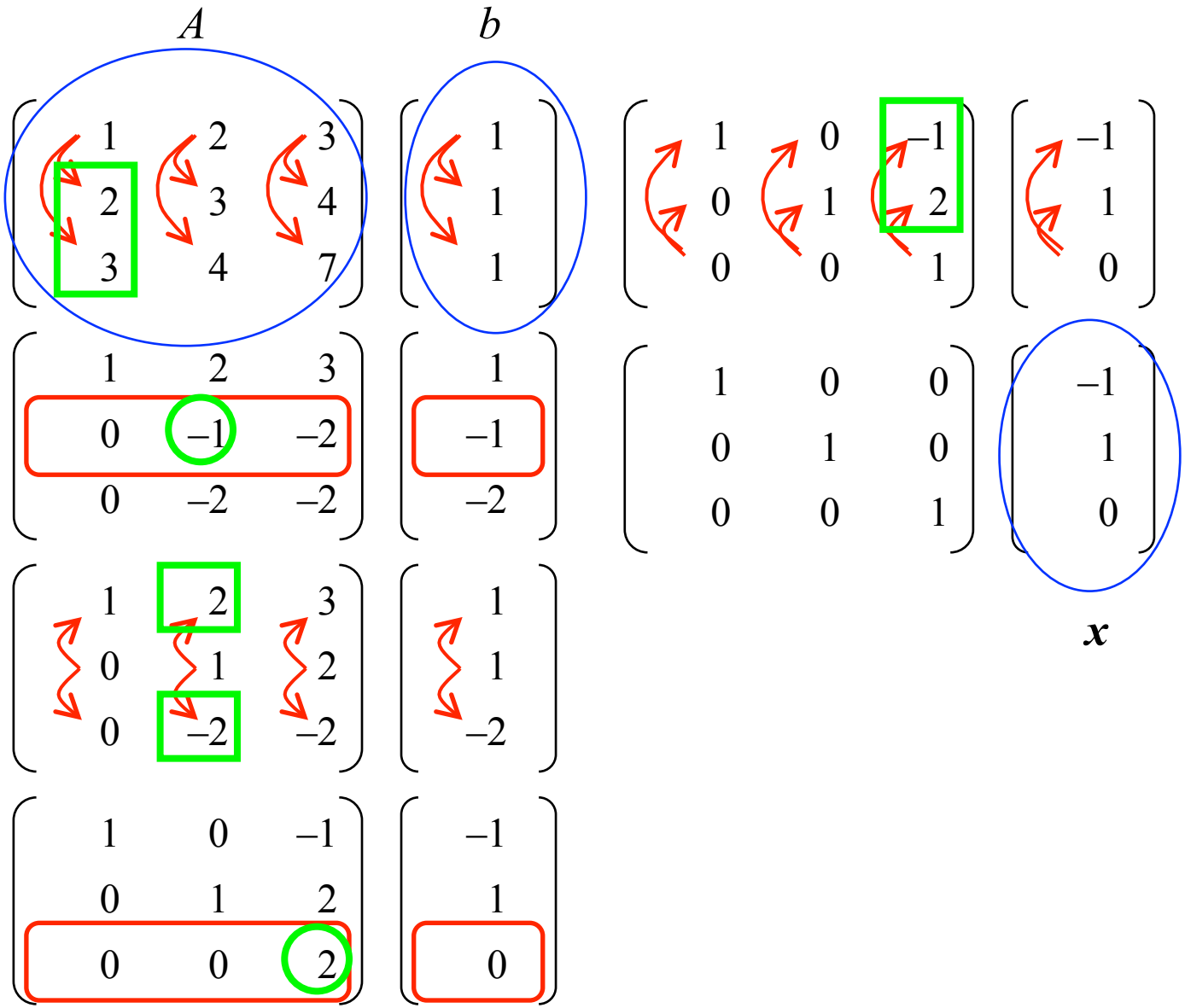
$$\boldsymbol{x} = E_kE_{k-1}\cdots E_2E_1\ b$$

I を目指して
A に基本行演算を
行っていく

同じ基本行演算を
b に対して
行っていく

4. ガウス-ジョルダン法（掃出し法による連立一次方程式の解法）

- 例



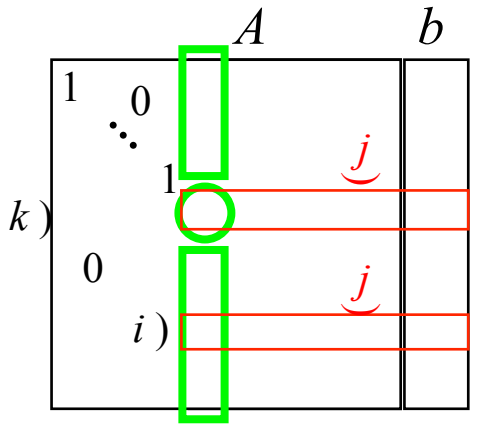
4. ガウス-ジョルダン法（掃出し法による連立一次方程式の解法）

C言語プログラム

逆行列の場合とは
ここが違うだけ

```
m = n + 1;
for (k=0; k<n; k++){
  for (j=k+1; j<m; j++)
    Ab[k][j] /= Ab[k][k];
  Ab[k][k] = 1.0;
  for (i=0; i<n ; i++){
    if (i==k) continue;
    for (j=k+1; j<m; j++)
      Ab[i][j] -=
        Ab[i][k] * Ab[k][j];
    Ab[i][k] = 0.0;
  }
}
```

[A | b] をひとつの配列で表し，
その要素を
Ab[k][j] (k =0,...,n–1, j =0,...,m–1)
とする



対角要素を1にする
（基本行演算「行のスカラー倍」を用いる）

非対角要素を 0 にする.
（基本行演算「ある行のスカラー倍を別の行に加える」を用いる. ）

4. ガウス-ジョルダン法（掃出し法による連立一次方程式の解法）

- 注意
  - 計算の手間を考慮するだけならば，ガウス消去法等を用いるべき
    - 連立一次方程式を解くだけなら，逆行列を求める必要はない
    - 逆行列を求める方法と同じコストのガウス-ジョルダン法も必要 ない

4. ガウス-ジョルダン法（掃出し法による連立一次方程式の解法）

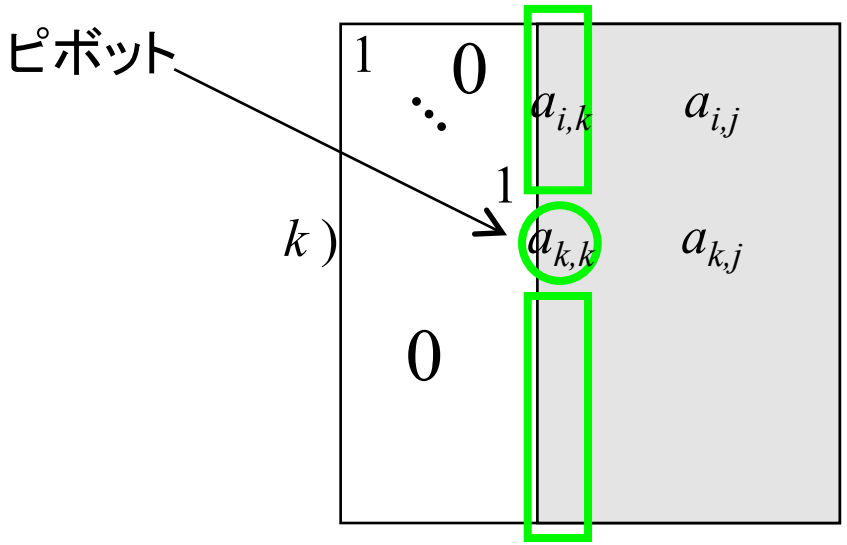
- 演習
  - 次の連立方程式を解け

$$\begin{pmatrix} 2 & 3 & 3 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} 2 \\ -3 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 2 & 3 \\ 2 & 3 & 2 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} -1.2 \\ 1.4 \\ 0.6 \end{pmatrix}$$

5. ピボット選択法

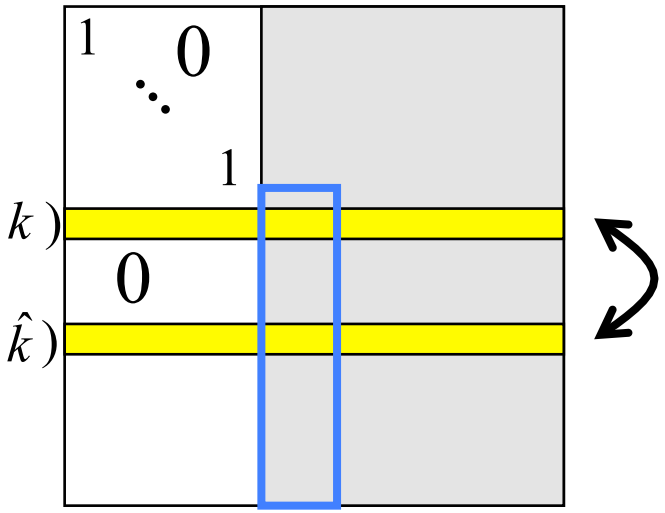
- 前述のアルゴリズムの問題点
  - $a_{k,k} = 0$ ⇒ゼロ除算が起こる
  - $a_{k,k}$ が非常に小さい ⇒
    - $a_{k,j} := a_{k,j}/a_{k,k}$ ⇒$a_{k,j}$ の持つ誤差の拡大
      ⇒$a_{k,j}$ 自身の値も大きくなる
    - $a_{i,j} := a_{i,j} - a_{i,k}\, a_{k,j}$ ⇒情報落ち（$a_{k,j}$が非常に大きいため）



5. ピボット選択法

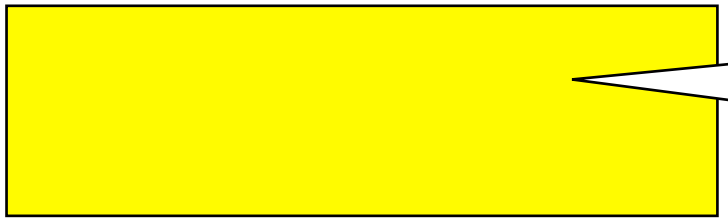
- ピボット選択法（部分選択法）
  - 基本行演算のうち「**行の交換**」はまだ使っていなかった.

$$E_{\hat{k},k} : \hat{k} = \underset{i=k,\ldots,n}{\arg\max} \mid a_{i,k} \mid$$



5. ピボット選択法

```
for (k=0; k<n; k++){
```



$\hat{k}$ を求め，k行と$\hat{k}$ 行を交換するプログラムをここに追加する

```
  for (j=k+1; j<m; j++)
    Ab[k][j] /= Ab[k][k];
  Ab[k][k] = 1.0;
  for (i=0; i<n ; i++){
    if (i==k) continue;
    for (j=k+1; j<m; j++)
      Ab[i][j] -=
        Ab[i][k] * Ab[k][j];
    Ab[i][k] = 0.0;
  }
}
```