C プログラミング

— 連立 1 次方程式の解法 —

早稲田大学

本日の目標

- 連立一次方程式の解法
  - ガウスの消去法
    - 前進消去
    - 後退代入
  - NaN・INFINITY
  - 部分ピボット選択付きガウスの消去法

連立 1 次方程式の解法

- 逆行列を用いる方法
- クラメール (Cramer) の公式
- 直接解法（消去法とも呼ばれる）: 密行列の計算に向く.
  - ガウス (Gauss) の消去法
  - LU 分解
- 反復解法 : 大規模な疎行列の解法に向く.
  - ヤコビ (Jacobi) 法
  - ガウス・ザイデル (Gauss-Seidel) 法

ガウスの消去法

例題 1

次の 3 元 1 次連立方程式をガウスの消去法を用いて解け.

$$\begin{cases} 2x & +4y & +6z & = & 6 \\ 3x & +8y & +7z & = & 15 \\ 5x & +7y & +21z & = & 24 \end{cases}$$

ガウスの消去法

前進消去

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ 3x & +8y & +7z & = & 15 & \cdots (2) \\ 5x & +7y & +21z & = & 24 & \cdots (3) \end{cases}$$

$\Downarrow$ $(2) - (1) \times \frac{3}{2}$ 【(2) から x を消去する】

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ 0x & +2y & -2z & = & 6 & \cdots (4) \\ 5x & +7y & +21z & = & 24 & \cdots (3) \end{cases}$$

$\Downarrow$ $(3) - (1) \times \frac{5}{2}$ 【(3) から x を消去する】

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ 0x & +2y & -2z & = & 6 & \cdots (4) \\ 0x & -3y & +6z & = & 9 & \cdots (5) \end{cases}$$

$\Downarrow$ $(5) - (4) \times \frac{-3}{2}$ 【(5) から y を消去する】

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ 0x & +2y & -2z & = & 6 & \cdots (4) \\ 0x & +0y & +3z & = & 18 & \cdots (6) \end{cases}$$

後退代入

$\Downarrow$ $(6)/3$ 【(6) から z を求める】

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ 0x & +2y & -2z & = & 6 & \cdots (4) \\ & & z & = & 6 & \cdots (7) \end{cases}$$

$\Downarrow$ $[(4) - (7) \times (-2)]/2$ 【y を求める】

$$\begin{cases} 2x & +4y & +6z & = & 6 & \cdots (1) \\ & y & & = & 9 & \cdots (8) \\ & & z & = & 6 & \cdots (7) \end{cases}$$

$\Downarrow$ $[(1) - (8) \times 4 - (7) \times 6]/2$ 【x を求める】

$$\begin{cases} x & & & = & -33 & \cdots (9) \\ & y & & = & 9 & \cdots (8) \\ & & z & = & 6 & \cdots (7) \end{cases}$$

前進消去と後退代入の組み合わせにより連立 1 次方程式の解を求める方法をガウスの消去法という.

ガウスの消去法

前進消去

$$\begin{pmatrix} 2 & 4 & 6 \\ 3 & 8 & 7 \\ 5 & 7 & 21 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 15 \\ 24 \end{pmatrix}$$

$\Downarrow$ 1 行目から 0 行目 $\times \frac{3}{2}$ を引く

$$\begin{pmatrix} 2 & 4 & 6 \\ 0 & 2 & -2 \\ 5 & 7 & 21 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \\ 24 \end{pmatrix}$$

$\Downarrow$ 2 行目から 0 行目 $\times \frac{5}{2}$ を引く

$$\begin{pmatrix} 2 & 4 & 6 \\ 0 & 2 & -2 \\ 0 & -3 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \\ 9 \end{pmatrix}$$

$\Downarrow$ 2 行目から 1 行目 $\times \frac{-3}{2}$ を引く

$$\begin{pmatrix} 2 & 4 & 6 \\ 0 & 2 & -2 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \\ 18 \end{pmatrix}$$

後退代入

$\Downarrow$ 2 行目を 3 で割る

$$\begin{pmatrix} 2 & 4 & 6 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \\ 6 \end{pmatrix}$$

$\Downarrow$ 1 行目から 2 行目 $\times(-2)$ を引き，2 で割る

$$\begin{pmatrix} 2 & 4 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 9 \\ 6 \end{pmatrix}$$

$\Downarrow$ 0 行目から 1 行目 $\times 4$ と 2 行目 $\times 6$ を引き，2 で割る

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -33 \\ 9 \\ 6 \end{pmatrix}$$

前頁の連立方程式の変形を行列形式で書き直した．以後，この書き方で説明する．

前進消去

- 連立 1 次方程式 $A\vec{x} = \vec{b}$ を変形して，行列 A を上三角行列の形にする.
- 0 列目について 1 行目以降の要素を 0 に，1 列目について 2 行目以降の要素を 0 に，という操作を繰り返す．今，$0, ..., k-1$ 列目の処理は済んでいるとして，k 列目の $k+1$ 行目以降を 0 にすることを考える.（次の例は $N = 5, k = 1$）

$$\begin{pmatrix} A_{0,0} & \underset{\downarrow k}{A_{0,1}} & A_{0,2} & A_{0,3} & A_{0,4} \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ 0 & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ 0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ 0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \Rightarrow \begin{pmatrix} A_{0,0} & \underset{\downarrow k}{A_{0,1}} & A_{0,2} & A_{0,3} & A_{0,4} \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ 0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\ 0 & 0 & A_{3,2} & A_{3,3} & A_{3,4} \\ 0 & 0 & A_{4,2} & A_{4,3} & A_{4,4} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

勿論，変形の前後で A と $\vec{b}$ の値は変化する（同じ記号を使っているので注意）.

前進消去

- 更に k 列目の中の $k+1, ..., i-1$ 行目の処理は済んでいるとして，i 行目の処理を考える．（次の例は $N=5, k=1, i=3$）

$$
\begin{array}{c}
\\
\\
\\
i \rightarrow \\
\\
\end{array}
\begin{array}{c}
\downarrow k \\
\begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} \\
0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\
0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4}
\end{pmatrix}
\end{array}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
\Rightarrow
\begin{array}{c}
\downarrow k \\
\begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} \\
0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\
0 & 0 & A_{3,2} & A_{3,3} & A_{3,4} \\
0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4}
\end{pmatrix}
\end{array}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
$$

前進消去

- $A_{3,1}$ を 0 にするには，3 行目から 1 行目の $\frac{A_{3,1}}{A_{1,1}}$ 倍を引けばよい.

$$
\begin{array}{cc}
 & \downarrow k \Rightarrow \\
i \rightarrow & \\
\Downarrow & \\
 & \uparrow j \Rightarrow
\end{array}
\begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} \\
0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\
0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4}
\end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
$$

$r = \frac{A_{3,1}}{A_{1,1}}$ として $j = 1, ..., 4$ について $A_{3,j} \leftarrow A_{3,j} - A_{1,j}r$, $b_3 \leftarrow b_3 - b_1 r$ とすればよい.
これを一般化すると，$r = \frac{A_{i,k}}{A_{k,k}}$ として $j = k, ..., N-1$ について
$A_{i,j} \leftarrow A_{i,j} - A_{k,j}r$, $b_i \leftarrow b_i - b_k r$ とすればよい.

前進消去

プログラムの概形は次のような 3 重ループになる：

```
for (k = 0; k < N - 1; k++) {     /* k=0,...,N-2 */
  for (i = k + 1; i < N; i++) {  /* i=k+1,...,N-1 */
    r = A[i][k]/A[k][k];
    for (j = k; j < N; j++) {     /* j=k,...,N-1 */
      A[i][j] -= A[k][j]*r;
    }
    b[i] -= b[k]*r;
  }
}
```

後退代入

- 連立 1 次方程式 $A\vec{x} = \vec{b}$ を更に変形して，A を上三角行列から単位行列の形にしたい．そうすれば，$\vec{x} = \vec{b}$ となり，$\vec{b}$ が答になる．
- 必要なのは $\vec{b}$ の値だけなので，実際のアルゴリズムでは A の値の更新は省略される．
- $N-1, ..., i+1$ 行目の処理は済んでいるとして，i 行目の処理を考える．(次の例は $i = 2$)

$$
\begin{array}{c}
\\
\Uparrow \\
i \rightarrow
\end{array}
\begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} \\
0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
$$

$$
\uparrow j \Rightarrow
$$

$b_2 \leftarrow (b_2 - A_{2,3}b_3 - A_{2,4}b_4)/A_{2,2}$ とすればよい．

これを一般化すると $b_i \leftarrow \left(b_i - \sum_{j=i+1}^{N-1} A_{i,j}b_j\right) \Big/ A_{i,i}$ とすればよい．

後退代入

プログラムは次のような 2 重ループになる.

```
for (i = N - 1; i >= 0; i--) {   /* i=N-1,...,0 */
  for (j = i + 1; j < N; j++) {  /* j=i+1,...,N-1 */
    b[i] -= A[i][j]*b[j];
  }
  b[i] /= A[i][i];
}
```

例題 1

例題 1

次元 N を入力した後，$N \times N$ 行列 A と N 次元ベクトル $\vec{b}$ のデータを要素ごとにキーボードから入力し，その行列・ベクトルを使って，連立 1 次方程式 $A\vec{x} = \vec{b}$ をガウスの消去法で解き，その計算過程を表示せよ．

- 表示は次のようにすること．

```
N= 3 【Enter】
A[0][0] = 2.0 【Enter】
A[0][1] = 4.0 【Enter】
A[0][2] = 6.0 【Enter】
A[1][0] = 3.0 【Enter】
A[1][1] = 8.0 【Enter】
A[1][2] = 7.0 【Enter】
A[2][0] = 5.0 【Enter】
A[2][1] = 7.0 【Enter】
A[2][2] = 21.0 【Enter】
b[0] = 6.0 【Enter】
b[1] = 15.0 【Enter】
b[2] = 24.0 【Enter】
```

```
A, b =
  2.00    4.00    6.00  |  6.00
  3.00    8.00    7.00  | 15.00
  5.00    7.00   21.00  | 24.00
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  6.00
  0.00   -3.00    6.00  |  9.00
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  6.00
  0.00    0.00    3.00  | 18.00

...
```

- 最後に b に解が入ることになる．

例題1のヒント（概形）

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

void PrintAb(double *A, double *b, int n){
  /*行列・ベクトルを表示する関数*/
}

int main(void) {
  int i, j, k, N;
  double r;
  double *ai, *ak;
  double *A, *b;

  printf("N=");
  scanf("%d",&N);

  A = (double *) malloc(N*N*sizeof(double));
  if(A==NULL){
    printf("Can't allocate memory.¥n");
    exit(1);
  }
  x = (double *) malloc(N*sizeof(double));
  if(x==NULL){
    printf("Can't allocate memory.¥n");
    exit(1);
  }
  ai = A;
  for( i = 0 ; i < N ; i++ ){
     for( j = 0 ; j < N ; j++ ){
        printf("A[%d][%d] = ", i , j );
        scanf("%lf", ai+j );
     }
     ai+=N;
  }

  for( i = 0 ; i < N ; i++ ){
     printf("b[%d] = ", i );
     scanf("%lf", b+i);
  }
  PrintAb(A, b, N);

  /*前進消去*/

  /*後退代入*/

  return 0;
}
```

例題1の解答

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

void PrintAb(double *A, double *b, int n){
  {
  int i, j;
  double *ai;
  printf("A,b=¥n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%6.1f¥t", *(ai+j));
    }
    printf("|%6.1f¥t", b[i]);
    ai+=n;
  }
  printf("¥n");
}

int main(void) {
  int i, j, k, N;
  double r;
  double *ai, *ak;
  double *A, *b;

  printf("N=");
  scanf("%d",&N);

  A = (double *) malloc(N*N*sizeof(double));
  if(A==NULL){
    printf("Can't allocate memory.¥n");
    exit(1);
  }
  x = (double *) malloc(N*sizeof(double));
  if(x==NULL){
    printf("Can't allocate memory.¥n");
    exit(1);
  }
```

```
  ai = A;
  for( i = 0 ; i < N ; i++ ){
     for( j = 0 ; j < N ; j++ ){
        printf("A[%d][%d] = ", i , j );
        scanf("%lf", ai+j );
     }
     ai+=N;
  }

  for( i = 0 ; i < N ; i++ ){
     printf("b[%d] = ", i );
     scanf("%lf", b+i);
  }
  PrintAb(A, b, N);

  /*前進消去*/
  ai = A;
  ak = A;
  for (k = 0; k < N - 1; k++){
    for (i = k + 1; i < N; i++){
      r = *(ai+i*N+k) / *(ak+k);
      for (j = k; j < N; j++){
        *(ai+i*N+j) -= *(ak+j) * r;
      }
      b[i] -= b[k] * r;
    }
    PrintAb(A, b, N);
    ak+=N;
  }

  /*後退代入*/
  ai=ak;
  for (i = N - 1; i >= 0; i--) {
    for (j = i + 1; j < N; j++) {
      b[i] -= *(ai+j) * b[j];
    }
    b[i] /= *(ai+i);
    PrintAb(A, b,N);
    ai-=N;
  }

  return 0;
}
```

例題 1

- 実行結果は次のようになる.

```
A, b =
  2.00    4.00    6.00  |  6.00
  3.00    8.00    7.00  | 15.00
  5.00    7.00   21.00  | 24.00
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  6.00
  0.00   -3.00    6.00  |  9.00
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  6.00
  0.00    0.00    3.00  | 18.00
```

```
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  6.00
  0.00    0.00    3.00  |  6.00
A, b =
  2.00    4.00    6.00  |  6.00
  0.00    2.00   -2.00  |  9.00
  0.00    0.00    3.00  |  6.00
A, b =
  2.00    4.00    6.00  |-33.00
  0.00    2.00   -2.00  |  9.00
  0.00    0.00    3.00  |  6.00
```

例題 2

例題 2

次の値での連立方程式 $A\vec{x} = \vec{b}$ を解け．計算過程を表示せよ．

$$A = \begin{pmatrix} 2 & 4 & 1 & -3 \\ -1 & -2 & 2 & 4 \\ 4 & 2 & -3 & 5 \\ 5 & -4 & -3 & 1 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} 0 \\ 10 \\ 2 \\ 6 \end{pmatrix}$$

例題 2

- 実行結果は次のようになる.

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
 -1.00   -2.00    2.00    4.00  | 10.00
  4.00    2.00   -3.00    5.00  |  2.00
  5.00   -4.00   -3.00    1.00  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00   -6.00   -5.00   11.00  |  2.00
  0.00  -14.00   -5.50    8.50  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     inf     inf  |   inf
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
```

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |   nan
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
```

「nan」や「inf」とは何か？　何故，正しい解が計算されないのか？

例題 2

- 実行結果は次のようになる.

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
 -1.00   -2.00    2.00    4.00  | 10.00
  4.00    2.00   -3.00    5.00  |  2.00
  5.00   -4.00   -3.00    1.00  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00   -6.00   -5.00   11.00  |  2.00
  0.00  -14.00   -5.50    8.50  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     inf     inf  |   inf
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
```

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |   nan
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
```

「nan」や「inf」とは何か？　何故，正しい解が計算されないのか？

NaN・INFINITY

- double 型のような浮動小数点を扱う際，多くの計算機では IEEE754 と呼ばれる 2 進浮動小数点の規格に従って処理が行われる.
- IEEE754 では，計算過程において発生する可能性がある数の中で，通常の浮動小数点数では表現できない特別な値が定義されている.
  例: NaN，+INFINITY，−INFINITY

NaN・INFINITY

非数 NaN (Not a Number)

- $0/0, \sqrt{-1}$ などの無効演算の結果として生成される特殊な数で，`nan` と表示される．
- nan を含む計算の結果は nan となる．例: $1 + \mathrm{nan} = \mathrm{nan}$

無限大 INFINITY

- オーバーフロー（計算結果が表現できる最大の数を越えること）が起きた時に処理を続行可能とするために導入された特殊な数で，`inf` と表示される．
- 計算結果が表現できる最小の数（絶対値としては大きい数）を下回った場合もオーバーフローとなり，`-inf` と表示される．
- inf を含む計算の結果は，inf にならず浮動小数点数になることもある．
  例: $1/\mathrm{inf} = 0$

例題 2

- 何故正しい解が計算されないのか？
- 前進消去の第 1 列消去において，例題では $A_{1,1} = 0$ の場合，
  $r = \dfrac{A_{2,1}}{A_{1,1}}$ の計算で 0 での割算が発生し，r の計算結果が-inf となる.

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
 -1.00   -2.00    2.00    4.00  | 10.00
  4.00    2.00   -3.00    5.00  |  2.00
  5.00   -4.00   -3.00    1.00  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00   -6.00   -5.00   11.00  |  2.00
  0.00  -14.00   -5.50    8.50  |  6.00
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     inf     inf  |   inf
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
```

```
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   inf
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  | 10.00
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |  0.00
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
A, b =
  2.00    4.00    1.00   -3.00  |   nan
  0.00    0.00    2.50    2.50  |   nan
  0.00     nan     inf     inf  |   nan
  0.00     nan     nan     nan  |   nan
```

部分ピボット選択付きガウスの消去法

- 何故正しい解が計算されないのか？
- 前進消去の第 k 列消去において，$A_{k,k}$（例題では $A_{1,1}$）$= 0$ の場合，$r = \frac{A_{2,1}}{A_{1,1}}$ の計算で 0 での割算が発生し，計算結果が`-inf` となる．

$$
\begin{pmatrix} A_{0,0} & \overset{\downarrow k}{A_{0,1}} & A_{0,2} & A_{0,3} & A_{0,4} \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ 0 & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ 0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ 0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \Rightarrow \begin{pmatrix} A_{0,0} & \overset{\downarrow k}{A_{0,1}} & A_{0,2} & A_{0,3} & A_{0,4} \\ 0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ 0 & 0 & A_{2,2} & A_{2,3} & A_{2,4} \\ 0 & 0 & A_{3,2} & A_{3,3} & A_{3,4} \\ 0 & 0 & A_{4,2} & A_{4,3} & A_{4,4} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
$$

- 方程式の順番を入れ替えても解は変わらないので，行を入れ替えて $A_{1,1} \neq 0$ となるようにすればよい．
- 絶対値が 0 に近い値で割ると，一般に計算精度が悪くなるため，常に 1 行目と $A_{1,1}, A_{2,1}, A_{3,1}, A_{4,1}$ の中で絶対値が最大なものの行との入れ替えを行うことにすれば更によい．

部分ピボット選択付きガウスの消去法

部分ピボット選択

前進消去の第 k 列消去において，（$A_{k,k} \neq 0$ であっても常に）k 行目と $A_{k,k}, \ldots, A_{N-1,k}$ の中で絶対値が最大なものの行との入れ替えを行う．
即ち，k 行目と $m \equiv \max_{i=k,\ldots,N-1} |A_{i,k}|$ 行目を入れ替える．

- 例えば，$A_{1,1}, A_{2,1}, A_{3,1}, A_{4,1}$ がそれぞれ $0, 3, -4, 2$ だとしたら，1 行目と 3 行目を入れ替えてから，第 k 列消去を行う．

$$
\begin{array}{r}
 \\
k \to \\
 \\
m \to \\

\end{array}
\begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} \\
0 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
0 & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\
0 & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
0 & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4}
\end{pmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}
$$

k 行目と m 行目の入れ替えを行う．
勿論，A の行だけでなく，$\vec{b}$ の対応する要素も入れ替える．
列は入れ替えないので，$\vec{x}$ の要素は入れ替えてはならない．

- たまたま $k = m$ の場合は，入れ替え作業は不要．

配列内の最大要素の探索

要素数 N の double 型配列の何番目に最大値が入っているかを求めるプログラム

- 次のように探索する方法が一般的．例文内のコメントをよく理解すること．

```
#include <stdio.h>
#define N 5
int main(void) {
  int m, i;
  double A[N] = { 2, 1, 5, 3, 4 };

  m = 0;                          /* 仮に 0 番目が最大だとする */
  for (i = 1; i < N; i++) {       /* 1 番目から最後まで調べる */
    if (A[i] > A[m]) {            /* もし i 番目の方が大きいなら */
      m = i;                      /* i 番を記憶 */
    }
  }
  printf("A[%d] = %f is max\n", m, A[m]);

  return 0;
}
```

部分ピボット選択付きガウスの消去法

- プログラムの概形

```
/* 前進消去 */
for (k = 0; k < N - 1; k++) {
  /* 部分ピボット選択 */
  m = k;  /* 仮の最大番号 */
  for (i = k + 1; i < N; i++) {
    ... 省略...
  }
  printf("column %d: row %d is max\n", k, m);
  if (m != k) {  /* m ≠ k の時入れ替え */
    ... 省略（入れ替えの関数：swap）...
  }
  /* 第 k 列消去 */
  for (i = k + 1; i < N; i++) {
    for (j = k; j < N; j++) {
      ... 省略...
    }
  }
  PrintAb(A, b);
}
```

アルゴリズムの実装手順

重要

- まず $N = 5$ などと具体的かつ簡単な例で，自分が手作業でその操作ができるようになること.
- 次に繰り返し操作の部分を i, j といった変数を用いて一般化した式にする.
- $i = 0$ や $i = N - 1$ のように最初や最後の場合については，一般化したつもりでも例外処理を必要とすることが多いので，確認する.

本日のまとめ

- 連立一次方程式の解法
  - ガウスの消去法
    - 前進消去
    - 後退代入
  - NaN・INFINITY
  - 部分ピボット選択付きガウスの消去法