Intermediate Programming

— System of linear equations—

Waseda Univ.

Today's topic

- How to solve a system of linear equations
  - Gaussian elimination
    - Forward substitution
    - Backward substitution
  - NaN • INFINITY
  - Gaussian elimination with partial pivoting

Method for solving systems of linear equations

- The method using inverse of A
- The method using Cramer's rule
- Direct method: It is suitable for solving dense matrix.
  - Gaussian elimination
  - LU factorization
- Iterative method: It is suitable for solving sparse matrix.
  - Jacobi method
  - Gauss-Seidel method

Gaussian elimination

Example 1

Solve a system of linear equations using Gaussian elimination:

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

Gaussian elimination

Forward substitution

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

$\Downarrow$ $(2) - (1) \times \frac{3}{2}$: Eliminating x in (2)

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

$\Downarrow$ $(3) - (1) \times \frac{5}{2}$: Eliminating x in (3)

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

$\Downarrow$ $(5) - (4) \times \frac{-3}{2}$: Eliminating y in (5)

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

Backward substitution

$\Downarrow$ $(6)/3$: Solving z in (6)

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

$\Downarrow$ $[(4) - (7) \times (-2)]/2$: Solving y

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

$\Downarrow$ $[(1) - (8) \times 4 - (7) \times 6]/2$: Solving x

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

The method for solving a system of linear equations using forward substitution and backward substitution is called Gaussian elimination.

Gaussian elimination

Forward substitution

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

$\Downarrow$ $(2) - (1) \times \frac{3}{2}$: Eliminating x in (2)

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

$\Downarrow$ $(3) - (1) \times \frac{5}{2}$: Eliminating x in (3)

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

$\Downarrow$ $(5) - (4) \times \frac{-3}{2}$: Eliminating y in (5)

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

Backward substitution

$\Downarrow$ $(6)/3$: Solving z in (6)

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

$\Downarrow$ $[(4) - (7) \times (-2)]/2$: Solving y

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

$\Downarrow$ $[(1) - (8) \times 4 - (7) \times 6]/2$: Solving x

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

It was rewritten by matrix form. After that, we explain Gaussian elimination using this form.

Forward substitution

- It transforms A in $A\vec{x} = \vec{b}$ to an upper triangular matrix.
- It repeats that "the elements after the 1th row in the 0th column is made 0" and "the elements after the 2th row in the 1th column is made 0', and so on. In this case, we consider the process to make the element 0 after the $k+1$th row in the kth column $(N = 5, k = 1)$, as it has already completed that process of the $0, ..., k-1$th column.

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

Values of A and$\vec{b}$ become the different value before and after transformation.

Forward substitution

- In addition, we consider the process to make the element 0 in the ith row, as it has already completed that the process of the $k+1, ..., i-1$ row in the kth column. $(N = 5, k = 1, i = 3)$

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

Forward substitution

- For making $A_{3,1}$ 0, we should subtract the values of $\frac{A_{3,1}}{A_{1,1}}$ times of the 1st row from the 3rd row.

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

For $r = \frac{A_{3,1}}{A_{1,1}}$, in $j = 1, ..., 4$, it holds $A_{3,j} \leftarrow A_{3,j} - A_{1,j} r$, $b_3 \leftarrow b_3 - b_1 r$.

Generalizing it, for $r = \frac{A_{i,k}}{A_{k,k}}$, in $j = k, ..., N-1$, it holds $A_{i,j} \leftarrow A_{i,j} - A_{k,j} r$, $b_i \leftarrow b_i - b_k r$.

Forward substitution

We can write the program to compute forward substitution using a triple loop as the following:

```
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;
  }
}
```

Backward substitution

- We would like to transform the upper triangular matrix A into the identity matrix, for solving system of linear equations. Therefore, we have $\vec{x} = \vec{b}$ and it is an obtained answer.
- Since all that is required is the values of $\vec{b}$, the update of the value of A is omitted in the practical use.
- We consider the process in the ith row, as it has already completed that process of the $N-1, ..., i+1$th row. $(i = 2)$

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

$$
\uparrow j \Rightarrow
$$

We should compute $b_2 \leftarrow (b_2 - A_{2,3}b_3 - A_{2,4}b_4)/A_{2,2}$

Generalizing it, we have $b_i \leftarrow \left(b_i - \sum_{j=i+1}^{N-1} A_{i,j}b_j\right) \Big/ A_{i,i}$.

Backward substitution

We can write the program to compute backward substitution using double loop as the following:

```
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];
}
```

Example 1

Example 1

Write the program to solve a system of linear equations using Gaussian elimination. After inputting N (the size of A), input the elements of a problem A, b by keyboard and display the calculation process.

- Output example is the following:

```
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

...
```

- Note that finally we will obtain a solution in `b`.

Hint

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

void PrintAb(double *A, double *b, int n){
  /*Function to display a matrix and a vector*/
}

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);

  /*Forward substitution*/

  /*Backward substitution*/

  return 0;
}
```

Answer of example

```
#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);

  /*Forward substitution*/
  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;
  }

  /*Backward substitution*/
  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;
}
```

Example1

- The result is shown like the following:

```
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
```

Example 2

Example 2

Solve the following problem:

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

Example 2

- The result is shown like the following:

```
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
```

What are “nan” and “inf” ?. Why correct solution is not calculated ?

Example 2

- The result is shown like the following:

```
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
```

What are “nan” and “inf” ?. Why correct solution is not calculated ?

NaN • INFINITY

- For handling floating point numbers (float, double), many computers use the IEEE 754 standard.
- The IEEE 754 standard defines special representations for unexceptional operations. Example: NaN, +INFINITY, −INFINITY

NaN • INFINITY

NaN (Not a Number)

- A special value defined in IEEE 754 to signal an invalid result from a floating-point operation.
- A NaN is generated by operations such as $0/0, \sqrt{-1}$.
- The result of calculation including NaN generates NaN.
  Example: $1 + \mathrm{nan} = \mathrm{nan}$

INFINITY

- Overflow: Values have grown too large to be the representation ranges.
- Computers that follow the IEEE floating-point standard generates a special value called `inf` when overflow occurs.
- It generates `-inf`, when values have grown too small to be the representation ranges (as the absolute value, it is a large value).
- Infinity is also produced by operations like dividing by 0, eg. 1.0/0.0.
- Note that $1/\mathrm{inf} = 0$.

Example 2

- Why correct solution is not calculated ?
- Ans.: In the 1st column of forward substitution, division by 0 has occurred in $r = \frac{A_{2,1}}{A_{1,1}}$, when $A_{1,1} = 0$ in example 2. Therefore, the result of r is `-inf`.

```
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
```

Gaussian elimination with partial pivoting

- Why correct solution is not calculated ?
- In the kth column of forward substitution, division by 0 has occurred in $r = \frac{A_{2,1}}{A_{1,1}}$, when $A_{k,k}$ (in the example, $A_{1,1}$) $= 0$. Therefore, the result of r is `-inf`.

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

- In order to avoid division by 0, we replace the rows, since the solution does not change even if we interchange the equation.
- The accuracy will be worse when division by a value that absolute value is almost 0 occurs. We perform an interchange of the row where the absolute value is the largest.

Gaussian elimination with partial pivoting

Partial pivoting

In kth column substitution, (even if $A_{k,k} \neq 0$), we perform an interchange of the row where the absolute value is the largest. We interchange the kth column with $m \equiv \max_{i=k,\ldots,N-1} |A_{i,k}|$ column.

- For example, for $A_{1,1} = 0, A_{2,1} = 3, A_{3,1} = -4, A_{4,1} = 2$, we perform the k column substitution after we interchange the 1st row with the 3rd row.

$$
\begin{array}{r}
 \\
k \rightarrow \\
 \\
m \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 & 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}
$$

The interchange of the kth row and mth row. We must interchange the row of A and the corresponding element of $\vec{b}$.

- In case of $k = m$, we don't interchange it.

Searching the maximum value

A program to search where the maximum value is stored?

- The following is an example:

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

  m = 0;                          /*  It assumes 0th array is the maximum*/
  for (i = 1; i < N; i++) {       /* Checking from 1st to end */
    if (A[i] > A[m]) {            /*  If ith value is larger than jth value,*/
      m = i;                      /* ith value is stored */
    }
  }
  printf("A[%d] = %f is max\n", m, A[m]);

  return 0;
}
```

Gaussian elimination with partial pivoting

- Outline of Gaussian elimination with partial pivoting

```
/* Forward substitution */
for (k = 0; k < N - 1; k++) {
  /* Partial pivoting */
  m = k;
  for (i = k + 1; i < N; i++) {
    ...
  }
  printf("column %d: row %d is max\n", k, m);
  if (m != k) {  /* If m ≠ k,  interchange */
    ...swap function...
  }
  /* kth substitution */
  for (i = k + 1; i < N; i++) {
    for (j = k; j < N; j++) {
      ...
    }
  }
  PrintAb(A, b);
}
```

Implementation of this algorithm

Tips

- So, try specific examples, calculate them, and try to generalize the results.
- And next, you will generalize it.
- Sometimes, you need to perform exceptional procedures at the beginning or end, so please be careful.

Summary

- How to solve a system of linear equations
  - Gaussian elimination
    - Forward substitution
    - Backward substitution
  - NaN • INFINITY
  - Gaussian elimination with partial pivoting