TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO MÔN HỌC

GIẢI TÍCH SỐ

ĐỀ TÀI:

GIẢI HỆ PHƯƠNG TRÌNH BẰNG PHƯƠNG PHÁP KHỬ GAUSS JORDAN

GV hướng dẫn: **TS. Hà Thị Ngọc Yến**

Nhóm sinh viên thực hiện:

	Họ và tên	MSSV
1.	Đỗ Phương Khải	20185458
2.	Thân Thị Mơ	20185467
3.	Đặng Thị Ánh	20185433
4.	Điệp Quyền Thắng	20185476
5.	Phùng Mạnh Lâm	20185461
6.	Trịnh Công Sơn	20185474
7.	Lê Thị Nhi	20185470

Hà Nội, 2020

MỤC LỤC

1. Phân công nhiệm vụ.

Họ tên	Nhiệm vụ
Đỗ Phương Khải	So sánh phương pháp Gauss và Gauss Jordan, xây dựng slide thuyết trình
Thân Thị Mơ	Tìm và làm ví dụ minh họa.
Đặng Thị Ánh	Xây dựng lý thuyết phương pháp Gauss Jordan.
Điệp Quyền Thắng	Xây dựng thuật toán, xây dựng chương trình, tìm ví dụ, sửa đổi bổ sung thuật toán, chương trình.
Phùng Mạnh Lâm	Làm slide thuyết trình, tìm tài liệu tham khảo, xây dựng thuật toán
Trịnh Công Sơn	Xây dựng chương trình, thuyết trình
Lê Thị Nhi	Trình bày ví dụ minh họa, thuyết trình.

2. Quá trình báo cáo.

Ngày báo cáo: 08/05/2020.

Người thuyết trình: Trịnh Công Sơn, Lê Thị Nhi.

Câu hỏi:

Câu 1: Khi chạy chương trình trong trường hợp vô nghiệm, vô số nghiệm, giá trị nghiệm X hiện ra không chính xác, việc không tính được này thể hiện là hệ phương trình vô nghiệm, vô số nghiệm hay là do lỗi của chương trình?

Trả lời : Do chương trình nhóm em chưa được tối ưu khi cho vòng lặp vào sau mỗi lần tính nên có thể lưu lại giá trị của hệ phương trình trước đó và xảy ra tình trạng chượng trình chạy sai kết quả mong muốn. Nếu chạy độc lập mỗi hệ phương trình là một lần chạy chương trình thì vấn đề được giải quyết, nhưng sẽ tốn thao tác do chỉ xử lí được 1 hệ cho 1 lần chạy.

Câu 2: Khi thao tác trên code, có sử dụng đổi cột việc đó có ưu điểm và nhược điểm gì?

Trả lời:

- Ưu điểm: Hạn chế việc sử dụng các câu lệnh, dễ tính toán khi các phần tử khử đã được đưa về đường chéo chính.
- Nhược điểm: Sau khi đổi cột thi thứ tự nghiệm sẽ bị đổi chỗ cho nhau, Nếu không có thông báo cho người sử dụng biết thì sẽ dẫn đến kết quả bị sai.

Thành viên trả lời: Điệp Quyền Thắng.

Nhận xét:

-Nhận xét của giảng viên: Trong phương pháp Gauss-Jordan với giải hệ phương trình không nên đổi các cột với nhau vì sẽ phát sinh nhiều vấn đề, nhiều công việc cần giải quyết hơn. Nếu đổi thì phải đổi trả lại trong quá trình đọc nghiệm.

-Nhận xét của lớp: Không có.

3. Nội dung báo cáo.

3.1. Ý tưởng – Xây dựng công thức, thuật toán.

3.1.1 Ý tưởng phương pháp:

- Hạn chế sai số tính toán khi gặp các phép chia cho số gần 0 bằng cách chọn phần tử khử thích hợp.
- Sử dụng phương pháp khử(loại trừ ẩn) giống như giải hệ phương trình bằng phương pháp Gauss nhưng chúng ta không đưa phương trình đã cho về dạng ma trận tam giác, mà sẽ đưa hệ về dạng mỗi phương trình có 1 ẩn bằng các phép biến đổi đại số

3.1.2 Xây dựng công thức, thuật toán.

Gồm hai quá trình:

- Quá trình khử:

  Chọn phần tử trội hệ số a_{ij} phù hợp làm phần tử khử, tiến hành loại trừ ẩn.

  Xét hệ m phương trình n ẩn:

$$(1)\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1(n+1)} \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2(n+1)} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots. \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = a_{m(n+1)} \end{cases}$$

  Viết gọn: A.x=b. Ta viết Ab= A|b:

$$\begin{matrix} a_{11} & a_{12} \dots. & a_{1n} & a_{1(n+1)} \\ a_{21} & a_{22} \dots. & a_{2n} & a_{2(n+1)} \\ \dots & \dots & \dots & \dots \end{matrix}$$

$a_{m1} \quad a_{m2} \quad \dots \qquad a_{mn} \quad a_{m(n+1)}$

Ta gọi a_{pq} là phần tử giải nếu như:

Ưu tiên 1

| a_{pq}| = 1, 2, 4, 5 … (Để phép chia cho a_{pq} không có sai số hoặc sai số nhỏ).

Ưu tiên 2

| a_{pq}| =max | a_{ij}|, với i,j= 1,2,…,n.

Ta định nghĩa thêm dòng p và cột q lúc này là dòng giải và cột giải.

Giả sử Ab$^{(1)}$ = $a_{ij}^{(1)}$, với i,j=1,2,…,n.

Trong đó, $a_{pj}^{(1)} = a_{pj},\ với\, j = 1,2, \dots, n.$

$a_{ij}^{(1)} = \ a_{ij} - \frac{a_{pj}a_{iq}}{a_{pq}}$, mọi i,j và j ≠ p.

Như vậy, $a_{iq}^{(1)}$ sẽ đều bằng 0.

$a_{11} \quad a_{12} \dots. \quad 0 \quad \dots \quad a_{1n} \quad a_{1(n+1)}$

$a_{21} \quad a_{22} \dots. \quad 0 \quad \dots \quad a_{2n} \quad a_{2(n+1)}$

………………………………………..

$a_{p1} \quad a_{p2} \dots. \ a_{pq} \dots \quad a_{pn} \quad a_{p(n+1)}$

………………………………………..

$a_{m1} \quad a_{m2} \ \dots\ 0 \ \dots \quad a_{mn} \quad a_{m(n+1)}$

Bây giờ, ta lại đi tìm phần tử giải mới

Lặp lại quá trình trên, sau n – 1 bước, ta sẽ chọn

| $a_{kl}^{(1)}$| =max | $a_{ij}^{(1)}$|, với i,j= 1,2,…,n.

Trong đó, k ≠ p và l ≠ p.

Lặp lại quá trình như trên đến khi phần tử giải bằng 0 hoặc số lần lặp đạt m – 1.

Nếu phần tử giải bằng 0. Khi đó kiểm tra xem $a_{i(n+1)} \neq 0$ hay không ở các hàng i mà $a_{ij} = 0$,$\forall i = 1,2,\dots,n$. Nếu có, thông báo hệ vô nghiệm và kết thúc chương trình. Ngược lại chuyển sang quá trình đọc nghiệm.

Còn khi số lần lặp đạt m – 1 thì chuyển sang quá trình đọc nghiệm.

- Quá trình đọc nghiệm.

  Trong ma trận, những hệ số $a_{ij} \neq 0$ mà chưa từng là phần tử giải trong quá trình khử thì ta coi những nghiệm x_j như thế là nghiệm tham số (nghiệm biến đổi tự do).

  Sau đó thực hiện quá trình đọc nghiệm như trong phương pháp Gauss thì ta có nghiệm cần tìm.

3.2. Thuật toán.

Input: Ma trận A[m][n+1] (A|b)

Output: Nghiệm $x_1, x_2, x_3, \ldots, x_n$.

3.2.1. Quá trình khử.

- Bước 1:
  - Khởi tạo các giá trị x = y = 0, lưu giá trị tạm thời của phần tử khử.
  - Index[n]: Ghi lại vị trí các phần tử khử.
  - Used[m][n + 1] : Ma trận ghi lại các hàng và cột có phần tử khử.
- Bước 2: Nếu 1 < m, chuyển đến bước 3, ngược lại kết thúc quá trình khử.
- Bước 3: j = 1 → n . Nếu A[i][j] = 0 chuyển đến bước 4, ngược lại chuyển đến bước 5.
- Bước 4: Nếu A[i][n+1] != 0 thông báo hệ phương trình vô nghiệm và kết thúc chương trình, ngược lại chuyển đến bước 5.
- Bước 5: Nếu $|A_{ij}| = 1$, used[i][j] != 1 → x = i; y = i. Ngược lại chuyển đến Bước 6.
- Bước 6:
  - Nếu $|A_{ij}| = \max |A^{(k)}|$ khi Used[i][j] != 1, k = 1,2,….m.

- $max|A^{(k)}| = 0$ → kết thúc quá trình khử
  - Ngược lại x = i ; y = j.
- Bước 7: Gán giá trị index[i] = y.
- Bước 8: p = 1 → m; q = 1 → n
  - Nếu p = x hoặc q = y thì Used[p][q] = 1
- Bước 9: Lặp i = 1 → m
  - Nếu i = x → thực hiện lệnh kế tiếp
  - Tính mid = A[i][j] / A[x][y]
- Bước 10: Lặp i = 1 → m, j = 1 → n+1

  A[i][j] = A[i][j] – mid * A[x][j]
- Bước 11: Tìm max|A| với Used[i][j] != 1
- Bước 12: Tăng i = i + 1 và chuyển đến bước 2.

3.2.2. <u>Quá trình đọc nghiệm.</u>

- Bước 1: Khởi tạo Rind = 0: hạng của A

  RankM = 0: hạng của A|b
- Bước 2: Lặp i = 1→ n
  - Nếu index[i] != 0 thì Rind++
  - RankM = số hàng khác 0 của A|b sau quá trình khử.
- Bước 3: Nếu Rind != rankM thì hệ phương trình Vô nghiệm, ngược lại chuyển đến bước 4.
- Bước 4: Nếu Rind = n và Rind = rankM thìHệ phương trình có nghiệm duy nhất. Ngược lại chuyển đến bước 6.
- Bước 5: i → n
  - l = index[i]
  - X[i] = A[l][n+1] / A[l][i]
- Bước 6: Nếu Rind = RankM < n thì hệ phương trình có vô số nghiệm
- Bước 7: k = n – Rind, với k là số ẩn tham số.
- Bước 8: i = 1 → k,   j = 1 → n

  Nếu index[i] = 0 thì c[i] = j ; c1[i] = index[j].
- Bước 9: i = 1 → Rind, j = 1 →n

  Nếu index[j] != 0 thì v[i] = j ; v1[i] = index[j].
- Bước 10: j = 1 →k; i = 1 → n

  Nếu i = c[j] → X_i nhận giá trị bất kì ∈ R

- Bước 11: i = 1 → Rind ; j = 1 → n ; t = 1 → k
  Nếu j = v[i] → l = v[i] , l1 = v1[i]

$$x_j = \frac{A[l1][n+1]}{A[l1][j]} - \frac{A[l1][c(t)]}{A[l1][j]} * X_{c(t)}$$

3.3. Chương trình.

3.3.1. Chương trình chính.

```
main()
{

    FILE *fp;
    char FName[M];
    int m, n, x, y, i;
    int s = 0;
    int choice;
    int key;
    int Is = 0;
    int Count = 0;
    int index[M];
    double A[M][M],B[M][M];
    double Maxx = 0;
    double Solutions[M];
    int used[M][M];
    for( int i = 1; i <= m; ++i)
        for ( int j = 1; j <= n; j++)
            used[i][j] = 0;
    for( int i = 1; i <= n; ++i)
        index[i] = 0;

   do
   {
      printf( "\n\tGAUSS-JORDAN METHOD\n" );
      printf( "\t       (Ax=b)         \n" );
      printf( "1.Read Matrix from file.\n");
      printf( "2.Enter Matrix from keyboard.\n");
      printf( "0.Exit.\n");
      scanf( "%d", &key );
      switch(key)
      {
      case 1:
          printf("\nEnter the file name: ");
          scanf("%s", &FName );
          //gets(FName);
          fp = fopen( FName, "r" );
          if(!fp)
          {
              printf("\nFile ko ton tai.");
              break;
          }
          fscanf( fp, "%d\n", &m );
          fscanf( fp, "%d\n", &n );
```

```
    for( int i = 1; i <= m; ++i )
     for( int j = 1; j <= n+1; ++j )
     {
        fscanf( fp, "%lf\t", &A[i][j]);
     }
    fclose( fp );
    PrintEquations( A, m, n);
    Is = Isnotsolution( A, m,n,Is );
    printf("\n%d",Is);
    for( int i = 1; i <= m; ++i )
     for( int j = 1; j <= n+1; ++j )
     {
        B[i][j] = A[i][j];
     }
    for( int i = 1; i <= n; ++i)
         Solutions[i] = 0;
    tt:
     {
         while( Count < m-1 || Maxx != 0 )
         {
             Is = Isnotsolution( A, m,n,Is );
             if( Is == 1 )
             {
                 printf( "\nThe system of equations has no solution! ");
                 printf( "\nReason : Appearance as 0X1 + 0X2 +...+ Xn = b. ");
                 break;
             }
             x = 0; y =0;
             Maxx = Maxabs( A, used, m, n);
             x = Finex_element( A,used,Maxx, m, n, x);
             y = Finey_element( A,used,Maxx, m, n, y);
             //OrderMatrix(A,m,n,x,y);
             //y = x;
             i = y;
             index[i] = x;
             Mark(used, m, n, x, y);
             Handle_element( A, m, n, x, y);
             //printf("Reduction %d:",Count+1);
             //PrintMatrix(A,m,n);
             Maxx = Maxabs( A, used, m, n);
             //printf("\n");
             Count++;
         }
         if( Is == 0 )
         {
             printf("\nIndex of reducing element is:");
             for( int i = 1; i <= n; ++i)
                 printf("\t%d",index[i]);
             printf("\nThe Matrix after Gauss-Jordan method.");
             PrintMatrix(A,m,n);
             PrintResult(A,B,index,m,n);
             fflush(stdin);
         }
     }
     break;
case 2:
     printf( " Enter the number of variables:\n" );
     scanf( "%d", &n );
     printf( " Enter the number of  the equations:\n" );
     scanf( "%d", &m );
     EnterMatrix( A, m, n );
     PrintEquations( A, m, n );
```

```
            Is = Isnotsolution( A, m,n, Is );
            printf("\n%d",Is);
            for( int i = 1; i <= m; ++i )
                for( int j = 1; j <= n+1; ++j )
                {
                    B[i][j] = A[i][j];
                }
            for( int i = 1; i <= n; ++i)
                Solutions[i] = 0;
            goto tt;
            break;
        case 0:
            exit(0);
        default:
            printf("Try again.");
            break;
        }

    }while( key != 1 || key != 2 || key != 0);
}
```

3.3.2. Hàm tìm giá trị tuyệt đối lớn nhất.

```
double Maxabs( double A[M][M],int used[M][M], int m, int n )
{
    double Maxx=0;
    double tmp,t;
    for( int i = 1; i <= m; i++ )
        for( int j = 1; j <= n; j++)
        {
            if( used[i][j] == 1)
                continue;
            tmp = A[i][j];
            t = fabs(tmp);
            if( Maxx < t )
                Maxx = t;
        }
    return Maxx;
}
```

3.3.3. Hàm tìm vị trí phần tử khử.

```
int Finex_element( double A[M][M],int used[M][M], double Maxx, int m, int n,
int x)
{
    double tmp;
    for( int i = 1; i <= m; ++i )
    {
        for( int j = 1; j <= n; ++j )
        {
            tmp = A[i][j];
            /// Bỏ qua vị trí  hàng và cột đã khử.
            if(used[i][j] == 1)
                continue;
```

```
            if( fabs( tmp ) == 1 )
            {
                x = i;
                return x;
                break;

            }
        }
        if( x == 0 )
        {
            for( int j = 1; j <= n; ++j )
            {
                tmp = A[i][j];
                if(used[i][j] == 1)
                continue;
                else if( fabs(tmp) == Maxx )
                {
                    x = i;
                    return x;
                    break;

                }
            }
        }
    }

}

int Finey_element( double A[M][M],int used[M][M], double Maxx, int m, int n,
int y)
{
    double tmp;
    for( int i = 1; i <= m; ++i )
    {
        for( int j = 1; j <= n; ++j )
        {
            tmp = A[i][j];
            /// Bỏ qua vị trí  hàng và cột đã khử.
            if(used[i][j] == 1)
                continue;
            if( fabs( tmp ) == 1 )
            {
                y = j;
                return y;
                break;

            }
        }
        if( y == 0 )
        {
            for( int j = 1; j <= n; ++j )
            {
                tmp = A[i][j];
                if(used[i][j] == 1)
                continue;
                else if( fabs(tmp) == Maxx )
```

```
                {
                    y = i;
                    return y;
                    break;

                }
            }
        }
    }
}
```

3.3.4. Hàm lưu lại vị trí cột hàng có phần tử khử.

```
 int Mark( int used[M][M], int m, int n,int x, int y )
{
    for( int i = 1; i <= m; ++i )
        for( int j = 1; j <= n; ++j )
            if( i == x || j == y )
                used[i][j] = 1;
}
```

3.3.5. Hàm thực hiện quy trình khử.

```
 double Handle_element( double A[M][M], int m, int n,int x,int y )
{
    double mid;
    for( int i = 1; i <= m; ++i)
    {
        /// Giữ nguyên giá trị hàng thứ i.
        if( i == x )
            continue;
        mid = A[i][y]/A[x][y];
        for( int j = 1; j <= n + 1; ++j )
        {
            A[i][j] = A[i][j] - A[x][j] * mid;

        }
    }

}
```

3.3.6. Hàm kiểm tra phương trình vô nghiệm?

```
 int  Isnotsolution( double A[M][M], int m, int n, int Is)
{
    int dem = 0;
    for( int i = 1; i <= m; ++i )
    {
        for( int j = 1; j <= n; ++j )
        {
            if( A[i][j] == 0 )
            {
```

```
                dem++;
                //return dem;
            }
            else
                continue;
        }
        if( dem == n )
        {
            if( A[i][n+1] != 0 )
                Is = 1;
                return Is;
                break;
        }
        else
        {
            dem = 0;
            continue;
        }
    }
    return Is;
}
```

3.3.7. Hàm xác định hạng của ma trận A|b.

```
int Rank(double A[M][M], int m, int n)
{
    int i ,j , rank = 0, flag = 0;
    for(i = 1; i <= m; ++i)
    {
        for(j = 1; j <= n+1; ++j)
        {
            if(A[i][j] != 0)
                flag = 1;
        }
        if(flag == 1)
        {
            rank++;
            flag = 0;
        }
    }
    return rank;
}
```

3.3.8. Hàm thực hiện quá trình đọc nghiệm.

```
double PrintResult( double A[M][M],double B[M][M] , int index[M], int m, int
n )
{
    int Rind = 0;
    int rankM = Rank(A,m,n);
    printf("\nRankM = %d\n",rankM);
    double Solutions[n];
    /// xác định rank của A thông qua index[].
    /// nếu khác 0 thì rank +1.
```

```
for( int i = 1; i <= n; ++i )
{
    if( index[i] != 0)
        Rind++;
}
printf("\nRind = %d",Rind);
/// Hạng của A nhỏ hơn hạng A|b => vô nghiệm
if( Rind != rankM )
{
    printf("\nThe system of equations has no solution!");
}
/// r(A) số ẩn -> Nghiệm duy nhất.
if( Rind == rankM && Rind == n )
{
    printf("\nThe system of equations has unique solution!");
    for( int i = 1; i <= n; ++i )
    {
        int l = index[i];
        Solutions[i] = A[l][n+1]/A[l][i];
        printf("\nX%d = %lf", i, Solutions[i]);
    }
}
/// r(A) = r(A|b) -> vô số nghiệm.
if( Rind == rankM && Rind < n )
{
    /// Số ẩn tham số bằng n trừ đi số phần tử khử.
    int k = n - Rind;
    int c[k];
    int v[Rind];
    int v1[Rind];
    double ValP[k];
    double Val[Rind];
    int US[n];
    int U[n];
    for( int i = 1; i <= n; ++i)
    {
        US[i] = 0;
        U[i] = 0;
    }
    printf("\nThe system of equations has countless solutions!\n");

    /// xác định các ẩn tham số bằng cách xét mảng index.
    /// index[j] = 0 -> ẩn j là ẩn tham số .
    for( int i = 1 ; i <= k; ++i )
        for( int j = 1; j <= n; ++j)
        {
            if( index[j] == 0 && U[j] != 1)
                {
                    c[i] = j;
                    U[j] = 1;
                    break;
                }
        }
    /// xác định các ẩn phụ thuộc bằng cách xét mảng index.
    /// index[j] != 0 -> ẩn j là ẩn tham số.
    for( int i = 1 ; i <= Rind; ++i )
```

```
        for( int j = 1; j <= n; ++j)
        {
            if( index[j] != 0 && US[j] != 1)
            {
                    v[i] = j;
                    v1[i] = index [j];
                    US[j] = 1;
                    break;
            }
        }
    for( int i = 1 ; i <= Rind; ++i )
        printf("%d\t",v[i]);
    printf("\n");
    for( int i = 1; i <= Rind; ++i)
        printf("%d\t",v1[i]);

    /// Đọc nghiệm
    for( int j = 1; j <= k; ++j )
    {
        /// Ẩn tham số
        for( int i = 1; i <= n; ++i )
        {
            int p,q,e;
            p = c[j];
            if( i == p )
            {
                printf("\nX%d any value of R.",i);
            }
        }
    }
    /// Ẩn phụ thuộc
    for( int i = 1; i <= Rind; ++i )
    {
        int l = v[i];
        int l1 = v1[i];
        for( int j = 1; j <= n; ++j )
        {
            if( j == l)
            {
                double b;
                b = A[l1][n+1]/A[l1][l];
                printf("\nX%d = %lf",j,b);
                for( int t = 1; t <= k; ++t )
                {
                    double b1;
                    int p = c[t];
                    b1 = A[l1][p]/A[l1][l];
                    if( fabs(b1) != 0.00)
                    {
                        if( b1 < 0 )
                        {
                            printf( " + %lfX%d", fabs(b1), p );
                        }
                        else
                            printf(" - %lfX%d", b1, p);
                    }
```

```
                    }
                }
            }

        }
        /// Nếu có ẩn tham số thì thực hiện
        printf("\n");
        if( k > 0)
        {
            /// Nhận giá trị cho các ẩn thâm số
            for( int i = 1; i <= k; ++i )
            {
                int p = c[i];
                if ( p != 0 )
                {
                    printf("\nX%d = ",p);
                    scanf("%lf",&Solutions[p]);
                }
            }
            for( int i = 1; i <= Rind; ++i )
            {
                int l = v[i];
                int l1 = v1[i];
                for( int j = 1; j <= n; ++j )
                {
                    if( j == l)
                    {
                        Solutions[l] = A[l1][n+1]/A[l1][l];
                        for( int t = 1; t <= k; ++t )
                        {
                            double b1,b2;
                            int p = c[t];
                            b1 = A[l1][p]/A[l1][l];
                            b2 = Solutions[p];
                            Solutions[l] = Solutions[l] - (b1*b2);
                        }
                    }
                }
            }
            for( int i = 1; i <= n; ++i )
                printf( "\nX%d = %lf ", i, Solutions[i] );
        }
    }
    printf("\n");
    if( ( Rind == rankM && rankM == n ) || (Rind == rankM && rankM < n ) )
        Check( B, Solutions, m, n);
}
```

3.3.9. Hàm kiểm tra tính đúng của phương pháp.

```
double Check( double B[M][M],double Solutions[M], int m, int n)
{
    double b[m];
    for( int i = 1; i <= m; ++i )
```

```
        b[i] = 0;
    for( int i = 1; i <= m; ++i )
        for( int j = 1; j <= n; ++j )
        {
            b[i] = b[i] +( B[i][j]*Solutions[j] );
        }
    printf( "\nA*Solutions: ");
    for( int i = 1; i <= m; ++i )
        printf( "\n%lf ", b[i] );
}
```

3.4. Ví dụ minh họa.

3.4.1. Hệ phương trình có nghiệm duy nhất.

Giải hệ phương trình:

$$\begin{cases} x_1 + 3x_3 + 2x_4 = 2 \\ 2x_1 + x_2 + 3x_4 = 4 \\ 3x_1 + 2x_3 + 7x_4 = 7 \\ 7x_1 + 2x_2 + x_4 = 3 \end{cases}$$

Giải: Lập các ma trận $A^{(k)}$, các phần tử tính theo công thức: a_{ij}=a_{ij}- $\frac{a_{pj}.a_{iq}}{a_{pq}}$ (*)

$A^{(0)}$	$\begin{matrix} \boxed{1} & 0 & 3 & 2 \\ 2 & 1 & 0 & 3 \\ 3 & 0 & 2 & 7 \\ 7 & 2 & 0 & 1 \end{matrix}$	$\begin{matrix} 2 \\ 4 \\ 7 \\ 3 \end{matrix}$	Chọn phần tử trội $a_{pq} = 1$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{(1)}$	$\begin{matrix} 1 & 0 & 3 & 2 \\ 0 & \boxed{1} & -6 & -1 \\ 0 & 0 & -7 & 1 \\ 0 & 2 & -21 & -13 \end{matrix}$	$\begin{matrix} 2 \\ 0 \\ 1 \\ 11 \end{matrix}$	Chọn phần tử trội $a_{pq} = 1$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{(2)}$	$\begin{matrix} 1 & 0 & 3 & 2 \\ 0 & 1 & -6 & -1 \\ 0 & 0 & -7 & \boxed{1} \\ 0 & 0 & -9 & -11 \end{matrix}$	$\begin{matrix} 2 \\ 0 \\ 1 \\ -11 \end{matrix}$	Chọn phần tử trội $a_{pq} = 1$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).

$A^{(3)}$	1 0 17 0 0 1 -13 0 0 0 -7 1 0 0 $\boxed{-86}$ 0	0 1 1 0	Chọn phần tử trội $a_{pq} = -86$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{*(3)}$	1 0 0 0 0 1 0 0 0 0 0 1 0 0 -86 0	0 1 1 0	Tính theo công thức (*) và giữ nguyên hàng giải.

Từ đó suy ra hệ phương trình:

$$\begin{cases} x_1 = 0 \\ x_2 = 1 \\ x_4 = 1 \\ -86x_3 = 0 \end{cases}$$

Hay ta có nghiệm duy nhất của hệ phương trình (1) là:
X = (0, 1, 0, 1)
Kết quả chạy chương trình:

```
Select D:\C,C++\GaussJver1_5.exe
 Enter the number of  the equations:
4
        AX = b

1 0 3 2 2

2 1 0 3 4

3 0 2 7 7

7 2 0 1 3

The system equations just enterred is:
1.00X1 + 3.00X3 + 2.00X4 = 2.00.

2.00X1 + 1.00X2 + 3.00X4 = 4.00.

3.00X1 + 2.00X3 + 7.00X4 = 7.00.

7.00X1 + 2.00X2 + 1.00X4 = 3.00.

Index of reducing element is:   1       2       4       3
The Matrix after Gauss-Jordan method.
1.00    0.00    0.00    0.00    0.00
0.00    1.00    -0.00   0.00    1.00
0.00    0.00    0.00    1.00    1.00
0.00    0.00    -86.00  0.00    0.00

RankM = 4

Rind = 4
The system of equations has unique solution!
X1 = 0.000000
X2 = 1.000000
X3 = -0.000000
X4 = 1.000000

A*Solutions:
2.000000
4.000000
7.000000
3.000000
```

3.4.2. Hệ phương trình có vô số nghiệm.

Giải hệ phương trình:

$$\begin{cases} 3x_1 + 2x_2 + 5x_3 - 4x_4 = 6 \\ 5x_1 + x_2 - 8x_3 + 2x_4 = 0 \\ 2x_1 + 3x_2 - 3x_3 + 2x_4 = 4 \\ -4x_2 - 10x_3 + 4x_4 = -10 \end{cases}$$

Giải: Lập các ma trận $A^{(k)}$, các phần tử tính theo công thức: a_{ij}=a_{ij}- $\frac{a_{pj}.a_{iq}}{a_{pq}}$ (*)

$A^{(0)}$	$\begin{matrix} 3 & 2 & 5 & -4 \\ 5 & \boxed{1} & -8 & 2 \\ 2 & 3 & -3 & 2 \\ 0 & -4 & -10 & 4 \end{matrix}$	$\begin{matrix} 6 \\ 0 \\ 4 \\ -10 \end{matrix}$	Chọn phần tử trội $a_{pq} = 1$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{(1)}$	$\begin{matrix} -7 & 0 & 21 & -8 \\ 5 & 1 & -8 & 2 \\ -13 & 0 & 21 & -4 \\ 20 & 0 & \boxed{-42} & 12 \end{matrix}$	$\begin{matrix} 6 \\ 0 \\ 4 \\ -10 \end{matrix}$	Chọn phần tử trội $a_{pq} = -42$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{*(1)}$	$\begin{matrix} 3 & 0 & 0 & -2 \\ \frac{50}{42} & 1 & 0 & \frac{-12}{42} \\ -3 & 0 & 0 & 2 \\ 20 & 0 & -42 & 12 \end{matrix}$	$\begin{matrix} 1 \\ \frac{80}{42} \\ 0 \\ -10 \end{matrix}$	Rút gọn hàng thứ nhất được ma trận bên cạnh.
$A^{(2)}$	$\begin{matrix} \boxed{3} & 0 & 0 & -2 \\ 0 & 1 & 0 & \frac{32}{63} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -42 & \frac{76}{3} \end{matrix}$	$\begin{matrix} 1 \\ \frac{80}{42} \\ 0 \\ \frac{-50}{63} \end{matrix}$	Chọn phần tử trội $a_{pq} = 3$, giữ nguyên hàng giải và tính các phần tử theo công thức (*).
$A^{(3)}$	$\begin{matrix} 3 & 0 & 0 & -2 \\ 0 & 1 & 0 & \frac{32}{63} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -42 & \frac{76}{3} \end{matrix}$	$\begin{matrix} 1 \\ \frac{80}{42} \\ 0 \\ \frac{-50}{63} \end{matrix}$	Nếu chọn tiếp phần tử a_{pq} thì phần tử trội chỉ có thể chọn là 0(không thỏa mãn.). Nên dừng phương pháp Gauss Jordan tại đây và suy ra dạng nghiệm của nó.

Từ đó ta suy ra được dạng nghiệm của hệ phương trình vô số nghiệm:

$$\begin{cases} x_1 = \dfrac{1}{3} + \dfrac{2}{3}t \\ x_2 = \dfrac{95}{63} - \dfrac{32}{63}t \\ x_3 = \dfrac{25}{1323} - \dfrac{38}{63}t \\ x_4 = t \end{cases}$$

Kết quả chạy chương trình:

D:\C,C++\GaussJver1_5.exe

```
2 3 -3 2 4

0 -4 -10 4 -10

The system equations just enterred is:
3.00X1 + 2.00X2 + 5.00X3 - 4.00X4 = 6.00.

5.00X1 + 1.00X2 - 8.00X3 + 2.00X4 = 0.00.

2.00X1 + 3.00X2 - 3.00X3 + 2.00X4 = 4.00.

0.00X1 - 4.00X2 - 10.00X3 + 4.00X4 = -10.00.

Index of reducing element is:   1       2       4       0
The Matrix after Gauss-Jordan method.
3.00    0.00    0.00    -2.00   1.00
-0.00   1.00    -0.00   0.51    1.51
0.00    0.00    0.00    0.00    0.00
-0.00   0.00    -42.00  25.33   -16.67

RankM = 3

Rind = 3
The system of equations has countless solutions!
1       2       3
1       2       4
X4 any value of R.
X1 = 0.333333 + 0.666667X4
X2 = 1.507937 - 0.507937X4
X3 = 0.396825 + 0.603175X4

X4 = 1

X1 = 1.000000
X2 = 1.000000
X3 = 1.000000
X4 = 1.000000

A*Solutions:
6.000000
-0.000000
4.000000
-10.000000
```

3.4.3. <u>Hệ phương trình vô nghiệm.</u>

Giải hệ phương trình:

$$\begin{cases} x - z = 1 \\ 2x + 3y + z = 6 \\ 3x + 3y = 9 \end{cases}$$

Ta có ma trận:

$$\begin{bmatrix} \boxed{1} & 0 & -1 & 1 \\ 2 & 3 & 1 & 6 \\ 3 & 3 & 0 & 9 \end{bmatrix} \xLeftrightarrow[-3L_1+L_3\to L_3]{-2L_1+L_2\to L_2} \begin{bmatrix} \boxed{1} & 0 & -1 & 1 \\ 0 & \boxed{3} & 3 & 4 \\ 0 & 3 & 3 & 6 \end{bmatrix} \xLeftrightarrow{-L_2+L_3\to L_3} \begin{bmatrix} \boxed{1} & 0 & -1 & 1 \\ 0 & \boxed{3} & 3 & 4 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Suy ra ta được hệ phương trình:

$$\begin{cases} x - z = 1 \\ 3y + 3z = 4 \\ 0x + 0y + 0z = 2 \end{cases}$$

Hệ vô nghiệm.

Kết quả chạy chương trình:

```
D:\C,C++\GaussJver1_5.exe

        GAUSS-JORDAN METHOD
              (Ax=b)
1.Read Matrix from file.
2.Enter Matrix from keyboard.
0.Exit.
2
 Enter the number of variables:
3
 Enter the number of  the equations:
3
        AX = b

1 0 -1 1

2 3 1 6

3 3 0 9

The system equations just enterred is:
1.00X1 - 1.00X3 = 1.00.

2.00X1 + 3.00X2 + 1.00X3 = 6.00.

3.00X1 + 3.00X2 = 9.00.

Index of reducing element is:   1       2       0
The Matrix after Gauss-Jordan method.
1.00    0.00    -1.00   1.00
0.00    3.00    3.00    4.00
0.00    0.00    0.00    2.00

RankM = 3

Rind = 2
The system of equations has no solution!
```

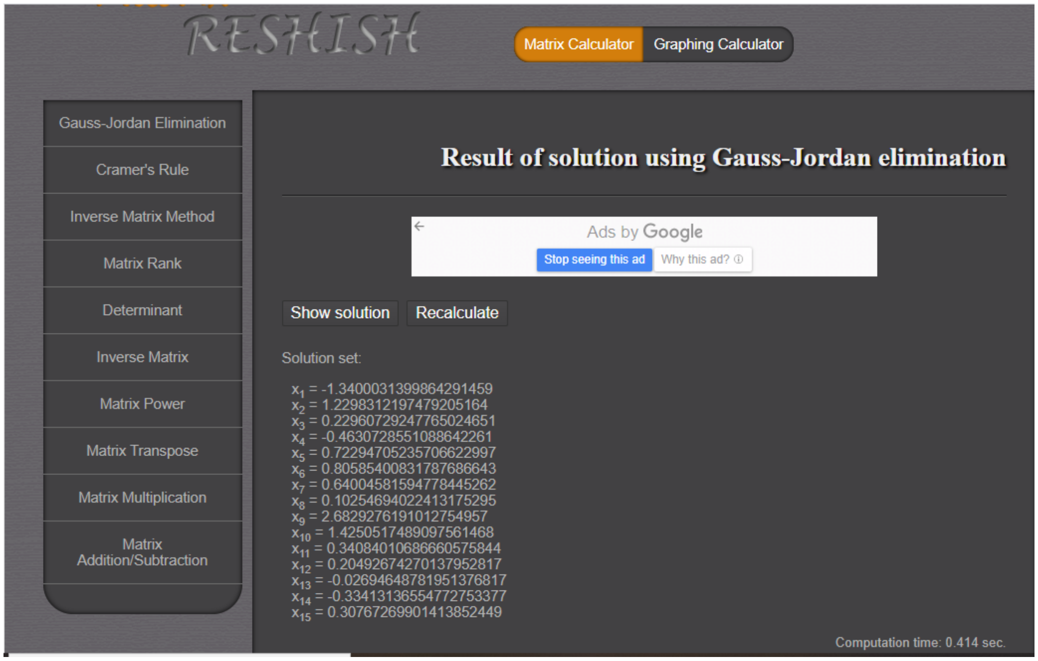
3.4.4. Nhận dữ liệu từ file:

Ở phần này chúng em chạy chương trình với hệ phương trình 15x15 nhận từ file MN1.txt

```
        GAUSS-JORDAN METHOD
              (Ax=b)
1.Read Matrix from file.
2.Enter Matrix from keyboard.
0.Exit.
1

Enter the file name: MN1.txt

The system equations just enterred is:
-7.00X1 + 39.00X2 - 38.00X3 + 7.00X4 + 86.00X5 + 45.00X6 - 81.00X7 - 7.00X8 - 99.00X9 + 99.00X10 - 83.00X11 + 55.00X12 + 68.00X13 + 97.00X14 - 47.00X15 = -99.00.

5.00X1 + 86.00X2 + 52.00X3 - 28.00X4 + 18.00X5 - 3.00X6 - 76.00X7 + 5.00X8 - 29.00X9 - 62.00X10 + 95.00X11 - 99.00X12 - 7.00X13 - 8.00X14 + 35.00X15 = -54.00.

-25.00X1 + 87.00X2 - 85.00X3 - 99.00X4 + 25.00X5 - 87.00X6 - 58.00X7 - 81.00X8 + 46.00X9 - 90.00X10 - 68.00X11 + 93.00X12 + 99.00X13 + 16.00X14 + 54.00X15 = 69.00.

18.00X1 - 24.00X2 - 28.00X3 + 82.00X4 + 91.00X5 - 11.00X6 - 95.00X7 - 12.00X8 + 6.00X9 + 28.00X10 + 3.00X11 + 66.00X12 + 19.00X13 + 19.00X14 - 18.00X15 = -45.00.

-30.00X1 - 18.00X2 + 18.00X3 - 92.00X4 + 10.00X5 + 51.00X6 + 24.00X7 + 3.00X8 - 20.00X9 - 54.00X10 - 29.00X11 + 15.00X12 + 39.00X13 + 82.00X14 + 49.00X15 = -22.00.

-5.00X1 - 29.00X2 + 41.00X3 - 45.00X4 - 99.00X5 + 66.00X6 - 24.00X7 + 42.00X8 - 6.00X9 + 28.00X10 + 49.00X11 - 70.00X12 - 75.00X13 + 36.00X14 = -12.00.

8.00X1 + 18.00X2 - 12.00X3 - 41.00X4 + 79.00X5 - 66.00X6 - 72.00X7 + 91.00X8 + 11.00X9 - 55.00X10 + 29.00X11 - 60.00X12 + 31.00X13 + 56.00X14 - 52.00X15 = -92.00.

-69.00X1 - 62.00X2 - 22.00X3 - 78.00X4 + 69.00X5 + 12.00X6 + 34.00X7 + 89.00X8 - 57.00X9 - 3.00X10 + 45.00X11 + 43.00X12 - 52.00X13 + 23.00X14 + 2.00X15 = -1.00.

16.00X1 - 91.00X2 - 71.00X3 - 36.00X4 - 53.00X5 + 40.00X6 + 82.00X7 + 35.00X8 + 27.00X9 - 18.00X10 + 29.00X11 - 73.00X12 - 71.00X13 + 53.00X14 - 68.00X15 = -78.00.

-63.00X1 + 31.00X2 + 99.00X3 + 23.00X4 - 65.00X5 - 87.00X6 + 41.00X7 - 2.00X8 - 78.00X9 + 42.00X10 - 12.00X11 - 54.00X12 + 96.00X13 - 43.00X14 + 30.00X15 = -100.00.

-33.00X1 + 56.00X2 - 100.00X3 + 89.00X4 - 36.00X5 + 18.00X6 + 19.00X7 - 90.00X8 + 15.00X9 - 27.00X10 + 55.00X11 + 98.00X12 - 96.00X13 + 95.00X14 - 48.00X15 = 37.00.

83.00X1 - 1.00X2 + 8.00X3 - 24.00X4 - 58.00X5 + 39.00X6 + 97.00X7 - 78.00X8 - 6.00X9 + 41.00X10 + 72.00X11 + 27.00X12 + 22.00X13 + 60.00X14 - 6.00X15 = -6.00.

-80.00X1 - 54.00X2 + 32.00X3 - 89.00X4 + 47.00X5 + 94.00X6 - 50.00X7 - 72.00X8 - 58.00X9 + 25.00X10 - 9.00X11 + 38.00X12 - 36.00X13 - 43.00X14 - 100.00X15 = 29.00.

-61.00X1 + 6.00X2 - 44.00X3 + 56.00X4 + 95.00X5 - 26.00X6 + 44.00X7 + 39.00X8 - 35.00X9 - 87.00X10 + 76.00X11 + 30.00X12 + 28.00X13 - 58.00X14 - 48.00X15 = -49.00.

-58.00X1 + 76.00X2 + 26.00X3 - 79.00X4 - 4.00X5 + 44.00X6 + 31.00X7 - 75.00X8 - 53.00X9 + 2.00X10 - 79.00X11 + 16.00X12 + 34.00X13 + 44.00X14 - 32.00X15 = 70.00.

Index of reducing element is:   1       2       3       4       5       6       7       8       9       10      11      12      13      14      15
```

```
The Matrix after Gauss-Jordan method.
189.73  0.00    -0.00   -0.00   0.00    0.00    0.00    -0.00   -0.00   0.00    0.00    0.00    0.00    -0.00   -0.00
        -254.24
0.00    -10.73  -0.00   0.00    0.00    -0.00   0.00    -0.00   -0.00   0.00    0.00    0.00    0.00    -0.00   -0.00
        -13.19
-0.00   -0.00   -120.53 0.00    -0.00   -0.00   -0.00   0.00    0.00    0.00    -0.00   -0.00   -0.00   0.00    -0.00
        -27.67
0.00    0.00    0.00    1482.22 0.00    -0.00   -0.00   0.00    -0.00   0.00    0.00    -0.00   -0.00   -0.00   0.00
        -686.38
0.00    0.00    -0.00   0.00    -33.18  0.00    0.00    -0.00   -0.00   0.00    0.00    0.00    0.00    -0.00   0.00
        -23.99
-0.00   0.00    0.00    0.00    -0.00   20.86   -0.00   0.00    -0.00   0.00    -0.00   -0.00   -0.00   -0.00   0.00
        16.81
0.00    -0.00   0.00    0.00    0.00    -0.00   -56.19  -0.00   -0.00   0.00    0.00    0.00    0.00    -0.00   0.00
        -35.96
-0.00   0.00    0.00    0.00    -0.00   -0.00   0.00    -72.96  0.00    -0.00   -0.00   0.00    0.00    0.00    -0.00
        -7.48
0.00    -0.00   -0.00   -0.00   0.00    -0.00   -0.00   -0.00   -67.08  0.00    0.00    0.00    0.00    -0.00   -0.00
        -179.98
-0.00   -0.00   0.00    0.00    -0.00   0.00    -0.00   0.00    0.00    107.42  -0.00   0.00    -0.00   0.00    -0.00
        153.08
-0.00   0.00    0.00    -0.00   -0.00   -0.00   -0.00   0.00    0.00    -0.00   55.00   -0.00   -0.00   0.00    0.00
        18.75
0.00    -0.00   -0.00   -0.00   0.00    -0.00   0.00    -0.00   -0.00   -0.00   0.00    894.05  0.00    -0.00   0.00
        183.21
-0.00   0.00    0.00    0.00    -0.00   -0.00   -0.00   0.00    0.00    0.00    -0.00   -0.00   603.68  -0.00   0.00
        -16.27
0.00    -0.00   -0.00   0.00    -0.00   0.00    0.00    0.00    -0.00   0.00    0.00    -0.00   0.00    -166.61 0.00
        55.67
-0.00   -0.00   -0.00   -0.00   -0.00   0.00    -0.00   0.00    0.00    -0.00   -0.00   0.00    -0.00   0.00    1618.53
        497.98
```

```
RankM = 15

Rind = 15
The system of equations has unique solution!
X1 = -1.340003
X2 = 1.229831
X3 = 0.229607
X4 = -0.463073
X5 = 0.722947
X6 = 0.805854
X7 = 0.640046
X8 = 0.102547
X9 = 2.682928
X10 = 1.425052
X11 = 0.340840
X12 = 0.204927
X13 = -0.026946
X14 = -0.334131
X15 = 0.307673

A*Solutions:
-99.000000
-54.000000
69.000000
-45.000000
-22.000000
-12.000000
-92.000000
-1.000000
-78.000000
-100.000000
37.000000
-6.000000
29.000000
-49.000000
70.000000
```



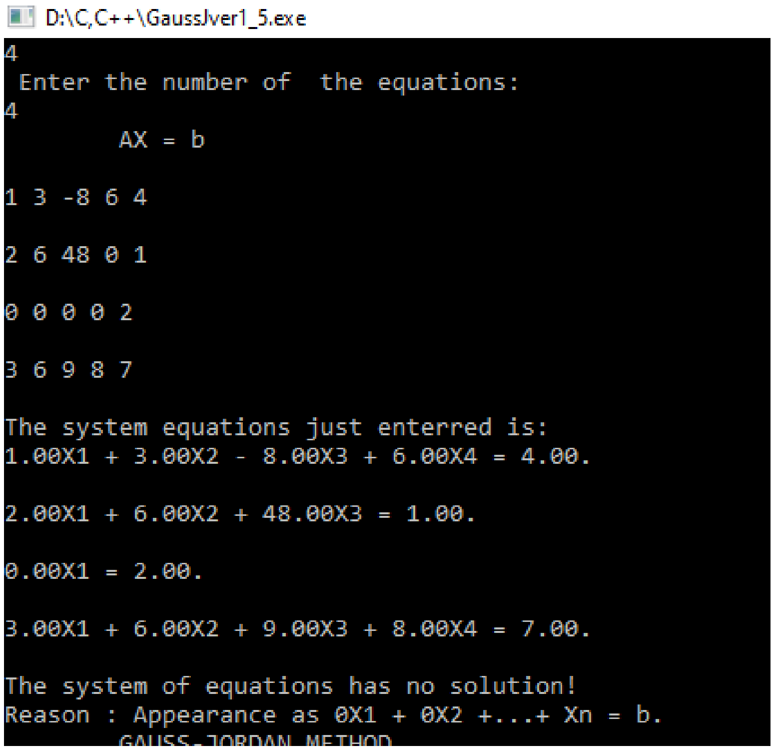
Kiểm tra lạ với trang https://matrix.reshish.com/

3.4.5. Một số ví dụ khác.

Ở phần này chúng em chỉ thực hiện chạy chương trình.

Ví dụ 1. Hệ vô nghiệm

$$\begin{cases} 1x_1 + 3x_2 - 8x_3 + 6x_4 = 4 \\ 2x_1 + 6x_2 + 48x_3 = 1 \\ 0x_1 + 0x_2 - 0x_3 + 0x_4 = 2 \\ 3x_1 + 6x_2 + 9x_3 + 8x_4 = 7 \end{cases}$$



```
D:\C,C++\GaussJver1_5.exe
4
 Enter the number of  the equations:
4
        AX = b

1 3 -8 6 4

2 6 48 0 1

0 0 0 0 2

3 6 9 8 7

The system equations just enterred is:
1.00X1 + 3.00X2 - 8.00X3 + 6.00X4 = 4.00.

2.00X1 + 6.00X2 + 48.00X3 = 1.00.

0.00X1 = 2.00.

3.00X1 + 6.00X2 + 9.00X3 + 8.00X4 = 7.00.

The system of equations has no solution!
Reason : Appearance as 0X1 + 0X2 +...+ Xn = b.
        GAUSS-JORDAN METHOD
```

Ví dụ 2: Hệ có nghiệm duy nhất.

$$\begin{cases} 1x_1 + 1x_2 - 3x_3 + 2x_4 = 6 \\ 1x_1 - 2x_2 - 1x_4 = -6 \\ 1x_2 + 1x_3 + 3x_4 = 16 \\ 2x_1 - 3x_2 + 2x_3 = 6 \end{cases}$$

D:\C,C++\GaussJver1_5.exe

```
 Enter the number of  the equations:
4
        AX = b

1 1 -3 2 6

1 -2 0 -1 -6

0 1 1 3 16

2 -3 2 0 6

The system equations just enterred is:
1.00X1 + 1.00X2 - 3.00X3 + 2.00X4 = 6.00.

1.00X1 - 2.00X2 - 1.00X4 = -6.00.

0.00X1 + 1.00X2 + 1.00X3 + 3.00X4 = 16.00.

2.00X1 - 3.00X2 + 2.00X3 = 6.00.

Index of reducing element is:   1       3       4       2
The Matrix after Gauss-Jordan method.
1.00    0.00    0.00    0.00    8.00
0.00    0.00    -0.00   0.92    1.85
0.00    1.00    0.00    0.00    6.00
0.00    0.00    13.00   0.00    52.00

RankM = 4

Rind = 4
The system of equations has unique solution!
X1 = 8.000000
X2 = 6.000000
X3 = 4.000000
X4 = 2.000000

A*Solutions:
6.000000
-6.000000
16.000000
6.000000
```

Ví dụ 3: Hệ có vô số nghiệm.

$$\begin{cases} 1x_1 + 2x_2 + 3x_3 + 4x_4 = 5 \\ 2x_2 + 3x_4 = 7 \\ 4x_3 + 6x_4 = 14 \\ 1x_3 + 4x_4 = 8 \end{cases}$$

```
D:\C,C++\GaussJver1_5.exe
The system equations just enterred is:
1.00X1 + 2.00X2 + 3.00X3 + 4.00X4 = 5.00.

0.00X1 + 2.00X3 + 3.00X4 = 7.00.

0.00X1 + 4.00X3 + 6.00X4 = 14.00.

0.00X1 + 1.00X3 + 4.00X4 = 8.00.

Index of reducing element is:   1       0       4       3
The Matrix after Gauss-Jordan method.
1.00    2.00    0.00    0.00    -4.60
0.00    0.00    0.00    0.00    0.00
0.00    0.00    -0.00   6.00    10.80
0.00    0.00    -1.67   0.00    -1.33

RankM = 3

Rind = 3
The system of equations has countless solutions!
1       3       4
1       4       3
X2 any value of R.
X1 = -4.600000 - 2.000000X2
X3 = 0.800000
X4 = 1.800000

X2 = 1

X1 = -6.600000
X2 = 1.000000
X3 = 0.800000
X4 = 1.800000

A*Solutions:
5.000000
7.000000
14.000000
8.000000
```

4. Đánh giá phương pháp và so sánh với các phương pháp.

4.1. Đánh giá phương pháp.

Phương pháp Gauss – Jordan là phương pháp giải đúng hệ phương trình tuyến tính

Có thể giải trong tất cả các trường hợp: nghiệm duy nhất, vô nghiệm và vô số nghiệm (Đưa ra dạng nghiệm).

Có thể giải với ma trận m < n.

Là phương pháp hạn chế sai số tính toán khi gặp các phép chia cho số gần 0.

4.2. So sánh với các phương pháp khác.

Phương pháp / Đặc điểm	**Gauss**	**Gauss - Jordan**	**Cholesky**
Điều kiện	Không	Không	Có
Độ phức tạp	O(n^3)	O(n^3)	O(n^3)
Nghiệm duy nhất	Có	Có	Có
Vô nghiệm	Có	Có	Không
Vô số nghiệm	Có	Có	Không
Nhận xét	Có thể giải được hệ bất kì. Không kiểm soát được sai số	Có thể giải được hệ phương trình bất ky. Hạn chế sai số	Ràng buộc đầu vào nên chỉ giải được những hệ thỏa mãn điều kiện. Giải nhanh hơn các phương pháp còn lại. Không kiểm soát được sai số.

5. Những cải tiến và bổ sung sau báo cáo.

Sau khi nhận được những nhận xét và góp ý của số trong buổi báo cáo ngày 08/05/2020, nhóm chúng em đã tiến hành những cải tiến và bổ sung sau:

5.1. Bỏ thao tác đổi cột trong chương trình

Trong quá trình báo cáo thì cô có góp ý rằng không nên đổi cột vì nó sẽ đổi cả thứ tự nghiệm và phát sinh nhiều vấn đề, nhiều công việc giải quyết hơn. Nên nhóm chúng em đã thực hiện bỏ thao tác đổi cột ngay trong buổi hôm đó và thực hiện chạy lại chương trình vào cuối buổi.

5.2. Sửa đổi và bổ sung cập nhật thuật toán.

Do có thực hiện bỏ và thêm thao tác vào chương trình nên cần phải sửa và bổ sung lại thuật toán:

- Sửa:
  - Tại quá trình khử: bỏ đổi cột x với cột y và gán y = x;
  - Tại quá trình đọc nghiệm:

    Bước 5 từ i → n, X[i] = A[i][n+1] / A[i][i] thành

    i → n, l = index[i], X[i] = A[l][n+1] / A[l][i]

    Bước 10 từ

    Nếu i = c[j] → X_i nhận giá trị bất kì ∈ R

    Nếu i != c[j] và A[i][j] = 0 → X_i = A[i][n+1] / A[i][i]

    Thành

    Nếu i = c[j] → X_i nhận giá trị bất kì ∈ R

    Bước 11 từ

    Nếu j = v[i] → $x_j = \frac{A[j][n+1]}{A[j][j]} - \frac{A[j][c(t)]}{A[j][j]} * X_{c(t)}$

    Thành

    Nếu j = v[i] → l = v[i] , l1 = v1[i]

$$x_j = \frac{A[l1][n+1]}{A[l1][j]} - \frac{A[l1][c(t)]}{A[l1][j]} * X_{c(t)}$$

- Bổ sung:
  - Tại quá trình khử:

    Thêm 2 bước vào trước bước 3.

    j = 1 → n . Nếu A[i][j] = 0 chuyển đến bước 4, ngược lại chuyển đến bước 5.

    Nếu A[i][n+1] != 0 thông báo hệ phương trình vô nghiệm và kết thúc chương trình, ngược lại chuyển đến bước 5.
  - Tại quá trình đọc nghiệm:

    Bước 9 thêm v1[i] = index[j].

5.3. Thêm chức năng nhận dữ liệu từ file và giải hệ phương trình.

Đối với những hệ phương trình có số lượng ẩn và số phương trình lớn thì việc nhập vào từ bàn phím sẽ mất rất nhiều thời gian. Cho nên chúng em thêm chức năng nhận dữ liệu từ file để tiết kiệm thao tác và thời gian

```
printf("\nEnter the file name: ");
            scanf("%s", &FName );
            //gets(FName);
            fp = fopen( FName, "r" );
            if(!fp)
            {
                printf("\nFile ko ton tai.");
                break;
            }
            fscanf( fp, "%d\n", &m );
            fscanf( fp, "%d\n", &n );

            for( int i = 1; i <= m; ++i )
             for( int j = 1; j <= n+1; ++j )
             {
                fscanf( fp, "%lf\t", &A[i][j]);
             }
            fclose( fp );
```

5.4. Thêm hàm kiểm tra hệ phương trình vô nghiệm?

Trước mỗi lần chọn phần tử khử ta sẽ xét xem hệ phương trình có phương trình nào có dạng $0x1+0x_2+\ldots+0x_n = b$.

Nếu có thì thông báo hệ vô nghiệm và kết thúc chương trình, ngược lại tiếp tục quá trình khử.

5.5. Thêm hàm nhân A với Solutions để kiểm tra lại nghiệm.

Sau khi thu được nghiệm của hệ ta sẽ tiến hành nhan ma trận hệ số A với vector nghiệm để kiểm tra tính đúng của phương pháp.

6. Tài liệu tham khảo.

Slide bài giảng, note của giảng viên cô Hà Thị Ngọc Yến.

Sách giải tích số của thầy Lê Trọng Vinh, xuất bản năm 2007, NXB Khoa học và Kĩ thuật.

Performance Comparison of Gauss Elimination and Gauss-Jordan Elimination, IJCCER.

Trang https://github.com/, https://matrix.reshish.com/gauss-jordanElimination.php .