Rappel +Algorithme+Pogramme en langage C

2.1-Elimination de Gauss-Jordan

L'élimination de Gauss-Jordan est un algorithme qui peut être utilisé pour résoudre des systèmes d'équations linéaires et pour trouver l'inverse de toute matrice inversible. Il repose sur trois opérations élémentaires de lignes que l'on peut utiliser sur une matrice:

1-Permuter les positions de deux des lignes
2-Multipliez l'une des lignes par un scalaire différent de zéro.
3-Ajoutez ou soustrayez le multiple scalaire d'une ligne à une autre ligne.

2.2-Exemple :

Pour un exemple de l'opération de première ligne élémentaire, permutez les positions de la 1ère et de la 3ème ligne.

$$\begin{bmatrix} 4 & 0 & -1 \\ 2 & -2 & 3 \\ 7 & 5 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 7 & 5 & 0 \\ 2 & -2 & 3 \\ 4 & 0 & -1 \end{bmatrix}$$

Pour un exemple de l'opération de deuxième ligne élémentaire, multipliez la deuxième ligne par 3.

$$\begin{bmatrix} 4 & 0 & -1 \\ 2 & -2 & 3 \\ 7 & 5 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 0 & -1 \\ 6 & -6 & 9 \\ 7 & 5 & 0 \end{bmatrix}$$

Pour un exemple de l'opération de troisième ligne élémentaire, ajoutez deux fois la 1ère ligne à la 2ème ligne.

$$\begin{bmatrix} 4 & 0 & -1 \\ 2 & -2 & 3 \\ 7 & 5 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 0 & -1 \\ 10 & -2 & 1 \\ 7 & 5 & 0 \end{bmatrix}$$

2.3-Étapes pour l'élimination de Gauss-Jordan :

Pour effectuer l'élimination de Gauss-Jordan:

1- Permutez les lignes de sorte que toutes les lignes avec toutes les entrées nulles soient en bas
2- Permutez les lignes de sorte que la ligne avec la plus grande entrée différente de zéro à gauche soit en haut.
3- Multipliez la ligne du haut par un scalaire pour que l'entrée principale de la ligne du haut devienne 1.
4- Ajoutez / soustrayez des multiples de la ligne supérieure aux autres lignes afin que toutes les autres entrées de la colonne contenant l'entrée principale de la ligne supérieure soient toutes nulles.
5- Répétez les étapes 2 à 4 pour la prochaine entrée différente de zéro à gauche jusqu'à ce que toutes les entrées principales soient 1.
6- Permutez les lignes de sorte que l'entrée principale de chaque ligne différente de zéro soit à droite de l'entrée principale de la ligne au-dessus.

Exemple :

https://www.youtube.com/watch?v=CsTOUbeMPUo

https://www.youtube.com/watch?v=PnIbv6cb4z0&list=RDCMUCcUf-yanjra_nDCWSh9U8Kw&index=5

Algorithme de Gauss-Jordan:

1. Start
2. Read the order of the matrix ‘n’ and read the coefficients of the linear equations.
3. Do for k=1 to n
   Do for l=k+1 to n+1
   a[k][l] = a[k][l] / a[k][k]

End for l
Set a[k][k] = 1
Do for i=1 to n
if (i not equal to k) then,
Do for j=k+1 to n+1
a[i][j] = a[i][j] – (a[k][j] * a[i][k])
End for j
End for i
End for k

4. Do for m=1 to n
   x[m] = a[m][n+1]
   Display x[m]
   End for m
5. Stop.

Programme de la méthode de Gauss-Jordan :

```
#include<stdio.h>
int main()
{
   int i,j,k,n;
   float A[20][20],c,x[10];
   printf("\nEnter the size of matrix: ");
   scanf("%d",&n);
   printf("\nEnter the elements of augmented matrix row-wise:\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=(n+1); j++)
      {
         printf(" A[%d][%d]:", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   /* Now finding the elements of diagonal matrix */
   for(j=1; j<=n; j++)
   {
      for(i=1; i<=n; i++)
      {
         if(i!=j)
         {
            c=A[i][j]/A[j][j];
            for(k=1; k<=n+1; k++)
            {
               A[i][k]=A[i][k]-c*A[j][k];
```

```
            }
          }
        }
    }
    printf("\nThe solution is:\n");
    for(i=1; i<=n; i++)
    {
       x[i]=A[i][n+1]/A[i][i];
       printf("\n x%d=%f\n",i,x[i]);
    }
    return(0);
}
```

3-La méthode de Gauss-seidel

https://www.youtube.com/watch?v=4h6zm5Rmf0A&list=RDCMUCcUf-yanjra_nDCWSh9U8Kw&index=3

4-La décomposition LU:

Voir https://www.youtube.com/watch?v=ZFnui4eR4j0&list=RDCMUCcUf-yanjra_nDCWSh9U8Kw&index=16

Algorithme de LU :

1. Start
2. Read the elements of augmented matrix into arrays a and b
3. Calculate elements of L and U
4. Print elements of L and U
5. Find V by solving LV = B by forward substitution
6. Find X by solving UX = V by backward substitution
7. Print Array X as the solution
8. Stop

Le programme de LU :

```
#include<stdio.h>
#include<conio.h>
void main()
{
   float A[20][20]= {0},L[20][20]= {0}, U[20][20];
   float B[20]= {0}, X[20]= {0},Y[20]= {0};
   int i,j,k,n;
   printf("Enter the order of square matrix: ");
   scanf("%d",&n);
   printf("\nEnter matrix element:\n");
   for(i=0; i<n; i++)
   {
      for(j=0; j<n; j++)
      {
         printf("Enter A[%d][%d] element: ", i,j);
         scanf("%f",&A[i][j]);
      }
   }
   printf("\nEnter the constant terms: \n");
   for(i=0; i<n; i++)
   {
      printf("B[%d]",i);
      scanf("%f",&B[i]);
   }
   for(j=0; j<n; j++)
   {
      for(i=0; i<n; i++)
      {
         if(i<=j)
         {
            U[i][j]=A[i][j];
            for(k=0; k<i-1; k++)
               U[i][j]-=L[i][k]*U[k][j];
            if(i==j)
               L[i][j]=1;
            else
               L[i][j]=0;
         }
         else
         {
            L[i][j]=A[i][j];
            for(k=0; k<=j-1; k++)
               L[i][j]-=L[i][k]*U[k][j];
            L[i][j]/=U[j][j];
            U[i][j]=0;
         }
```

```
        }
    }
    printf("[L]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",L[i][j]);
        printf("\n");
    }
    printf("\n\n[U]: \n");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%9.3f",U[i][j]);
        printf("\n");
    }
    for(i=0; i<n; i++)
    {
        Y[i]=B[i];
        for(j=0; j<i; j++)
        {
            Y[i]-=L[i][j]*Y[j];
        }
    }
    printf("\n\n[Y]: \n");
    for(i=0; i<n; i++)
    {
        printf("%9.3f",Y[i]);
    }
    for(i=n-1; i>=0; i--)
    {
        X[i]= Y[i];
        for(j=i+1; j<n; j++)
        {
            X[i]-=U[i][j]*X[j];
        }
        X[i]/=U[i][i];
    }
    printf("\n\n[X]: \n");
    for(i=0; i<n; i++)
    {
        printf("%9.3f",X[i]);
    }
    getch();
}
```