COMPUTACIÓN II

PRÁCTICA Nº 8: SISTEMAS DE ECUACIONES LINEALES. MÉTODOS ITERATIVOS.

(a) Escribir un programa para resolver un sistema de ecuaciones mediante los métodos iterativos de Jacobi y de Gauss-Seidel.

```
void jacobi(double A[n][n],double b[n],double newx[n], const int n)
void gaussseidel(double A[n][n], double b[n], double newx[n], const int n)
void lu(double A[n][n],double b[n],double x[n], const int n)
```

Se resolverá entonces el sistema:

$$\begin{bmatrix} -5 & 1 & 1 & 1 & 1 \\ 1 & -5 & 1 & 1 & 1 \\ 1 & 1 & -5 & 1 & 1 \\ 1 & 1 & 1 & -5 & 1 \\ 1 & 1 & 1 & 1 & -5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 14 \\ -10 \\ 4 \\ 6 \\ 12 \end{bmatrix}$$

CÓDIGO A

```
////////////////////////////////////////////////////////////////////////////
//Cristian Pinela Paniagua. 15/10/2013. Práctica 7-Método numérico//////////////
////////////////////////////////////////////////////////////////////////////
//Librerías
#include <iostream>
#include <cmath>
#include <fstream>

//Constantes generales
const int n=5;
const double tolerancia=1e-7;
```

```
//Funciones

float pe(double a[], double b[], int n);

float norma(double a[], int n);

void jacobi(double A[n][n], double b[n], double x[n], const int n);

void gaussseidel (double A[n][n], double b[n], double x[n], const int n);

void LU(double A[n][n], double b[n], double x[n], const int n);




using namespace std;


int main()

{

double A[n][n]={-5,1,1,1,1,1,-5,1,1,1,1,1,-5,1,1,1,1,1,-5,1,1,1,1,1,-5}; //Matriz de parámetros

double b[n]={14,-10,4,6,12}; //Matriz de soluciones

double x[n]; //Matriz de incógnitas

//Lo que tenemos que abordar en este problema es que al multiplicar A*x=b tenemos una matriz
cuyos //valores desconocemos, tenemos que averiguarla para cumplir la igualdad.

cout<<"La matriz A introducida es: "<<endl;

for (int i=0; i<n; i++) for (int j=0; j<n; j++)

{

cout<<A[i][j]<<" ";

if (j==n-1)

{cout<<endl;}

}

cout<<"Vemos que la matriz es diagonalmente dominante"<<endl;

cout<<endl<<"Y el espacio b de soluciones es: "<<endl;

for (int i=0; i<n; i++)

{

cout<<b[i]<<" "<<endl;

}

jacobi(A, b, x, n);

gaussseidel (A, b, x, n);

LU (A,b,x,n);
```

```
cout<<endl<<endl<<endl<<"Terminado!"<<endl;
//Líneas comunes en cada programa con dev c++
system ("PAUSE");
return 0;
}



//Cuerpo de las funciones
float pe(double a[], double b[], int n)
{
 float acu =0;

 for (int i = 0; i < n; i++)
 acu += a[i] * b[i];
 return acu;
}



float norma( double a[], int n)
{
 float aux;

 aux = pe(a, a, n);
 aux = sqrt(aux);
 return aux;
}



void jacobi(double A[n][n], double b[n], double x[n], const int n)
{
     double oldx[n];
     int c=0;

     //Vamos a hacer una primera aproximación de x.
```

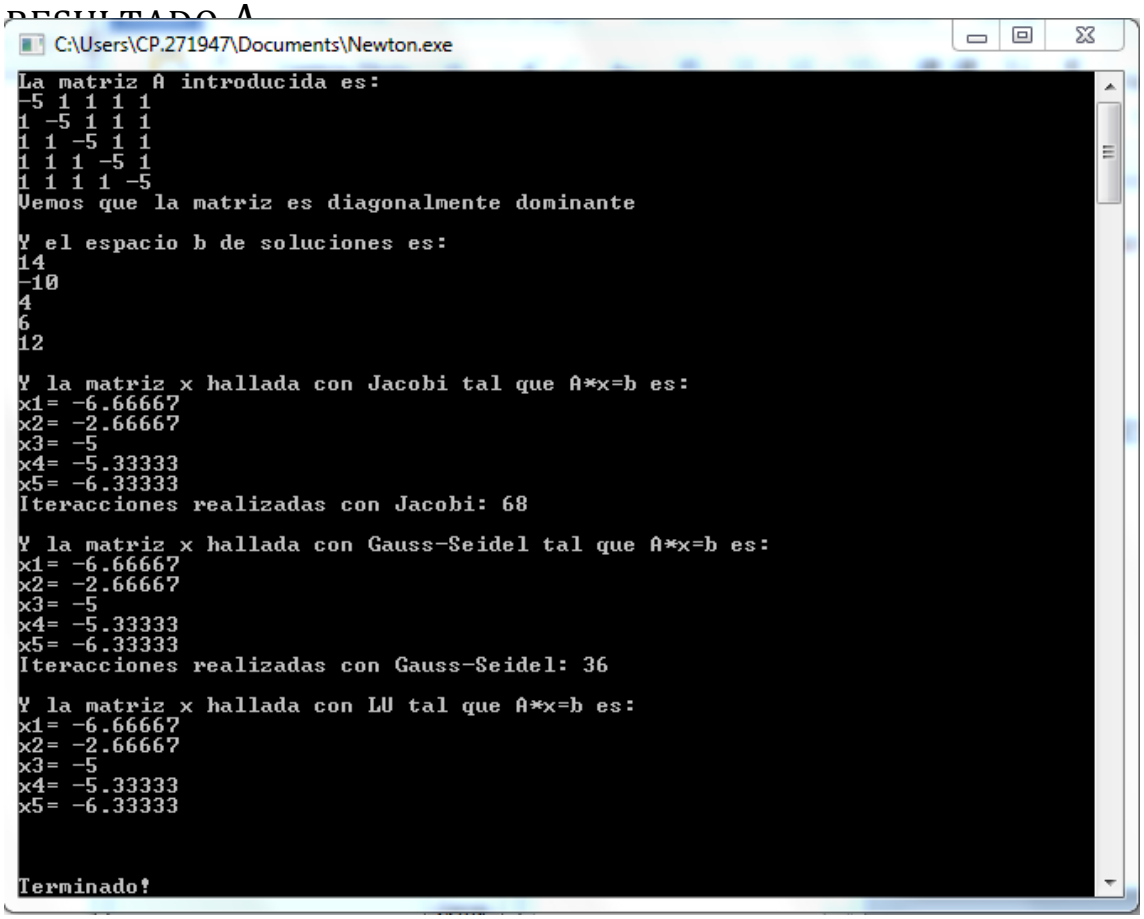
```
     for(int i=0; i<n; i++)
     {
             x[i]=b[i]/A[i][i];
     }
     do
     {
         for  (int i=0; i<n; i++)
         {
              oldx[i]=x[i];
         }
         for (int i=0; i<n; i++)
         {
             x[i]=b[i]/A[i][i];
             for (int j=0; j<n; j++)
             {
                 if (j!=i)
                 {
                 x[i]=x[i]-(A[i][j]/A[i][i])*oldx[j];
                 }
             }
         }
         c=c+1;
     }while ((fabs(norma(x, n)-norma(oldx,n)))>tolerancia);
     cout<<endl<<"Y la matriz x hallada con Jacobi tal que A*x=b es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<"x"<<i+1<<"= "<<x[i]<<endl;
}
     cout<<"Iteracciones realizadas con Jacobi: "<<c<<endl;
}

void gaussseidel (double A[n][n], double b[n], double x[n], const int n)
{
double oldx[n];
```

```
int c=0;


     //Vamos a hacer una primera aproximación de x.
     for(int i=0; i<n; i++)
     {
             x[i]=b[i]/A[i][i];
     }
     do
     {
         for  (int i=0; i<n; i++)
         {
              oldx[i]=x[i];
         }
         for (int i=0; i<n; i++)
         {
             x[i]=b[i]/A[i][i];
             for (int j=0; j<n; j++)
             {
                 if (j!=i)
                 {
                 x[i]=x[i]-(A[i][j]/A[i][i])*x[j];
                 }
             }
         }
         c=c+1;
     }while ((fabs(norma(x, n)-norma(oldx,n)))>tolerancia);
     cout<<endl<<"Y la matriz x hallada con Gauss-Seidel tal que A*x=b es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<"x"<<i+1<<"= "<<x[i]<<endl;
}
cout<<"Iteracciones realizadas con Gauss-Seidel: "<<c;


}
```

```
void LU(double A[n][n], double b[n], double x[n], const int n)
{
//Tenemos primero que definir dos matrices, la L y la U:
double L[n][n], U[n][n], z[n];
double sumU, sumL, sumz,sumx;
//Ahora que ya están definidas tenemos que darles valores, para eso utilizaremos un loop.
//Loops utilizados para hallar LU
for(int i=0; i<=n-1;i++)
for(int j=0; j<=n-1;j++)
{
//Caso general de L y U if (i<=j) //Condicional para hacer U
       {
         if (i<j) //Condicional del caso triángulo superior de L
            {L[i][j]=0;}
         if (i==j) //Condicional del caso diagonal de L
            {L[i][j]=1;}
         sumU=0;
         for(int k=0; k<=i-1; k++) //Loop con respecto a sumU
         {sumU=sumU+L[i][k]*U[k][j];}
         U[i][j]=A[i][j]-sumU;
         }
         if (i>j) //Condicional para hacer L
         {
         U[i][j]=0;
         sumL=0;
         for(int k=0; k<=j-1; k++) //Loop con respecto a sumL
         {sumL=sumL+L[i][k]*U[k][j];}
         L[i][j]=(A[i][j]-sumL)/U[j][j];
         }
}
//Loops utilizados para hallar z
for(int i=0; i<=n-1;i++) //Loop principal
{
```

```
        sumz=0;
        for(int j=0; j<=i-1;j++) //Loop utilizado para hallar el sumatorio de z
                {
                  sumz=sumz+L[i][j]*z[j];
                }
        z[i]=b[i]-sumz;
}
//Loops utilzados para hallar x
for(int i=n-1; i>=0; i--) //Loop principal
{
  double cx=0, sumx=0;
  for (int j=i+1; j<=n-1; j++) //Loop utilizado para hallar el sumatoriot
  {
  sumx=sumx+U[i][j]*x[j];
  }
  x[i]=(z[i]-sumx)/U[i][i];
}
cout<<endl<<endl<<"Y la matriz x hallada con LU tal que A*x=b es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<"x"<<i+1<<"= "<<x[i]<<endl;
}
}
```

RESULTADO A


C:\Users\CP.271947\Documents\Newton.exe
La matriz A introducida es:
-5 1 1 1 1
1 -5 1 1 1
1 1 -5 1 1
1 1 1 -5 1
1 1 1 1 -5
Vemos que la matriz es diagonalmente dominante
Y el espacio b de soluciones es:
14
-10
4
6
12
Y la matriz x hallada con Jacobi tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Iteracciones realizadas con Jacobi: 68
Y la matriz x hallada con Gauss-Seidel tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Iteracciones realizadas con Gauss-Seidel: 36
Y la matriz x hallada con LU tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Terminado!

(a) Escribir un programa para resolver un sistema de ecuaciones mediante los métodos iterativos de Jacobi y de Gauss-Seidel con punteros

CÓDIGO B

```
////////////////////////////////////////////////////////////////////////////
//Cristian Pinela Paniagua. 15/10/2013. Práctica 8-Método numérico///////////////
////////////////////////////////////////////////////////////////////////////
//Librerías
#include <iostream>
#include <cmath>
#include <fstream>

//Constantes generales
const double tolerancia=1e-7;



//Funciones
float pe(double a[], double b[], int n);
float norma(double a[], int n);
void jacobi(double **A, double *b, double *x, int n);
void gaussseidel (double **A, double *b, double *x, int n);
void LU(double **A, double *b, double *x, int n);



using namespace std;

int main()
{
double **A; //Matriz de incógnitas
double *x;
double *b;
```

```
int n=5;

//Lo que tenemos que abordar en este problema es que al multiplicar A*x=b tenemos una matriz
cuyos //valores desconocemos, tenemos que averiguarla para cumplir la igualdad.



cout<<endl;

cout<<endl<<endl;

//Creamos los punteros de los vectores

b= new double[n];

x= new double[n];

//Ahora en la matriz

A=new double* [n];

for(int i=0 ; i<n ;i++)

{A[i]=new double[n];}


//Vamos a rellenar la matriz A

cout<<"Vamos a rellenar la matriz A: "<<endl;

for (int i=0; i<n; i++)

for (int j=0; j<n; j++)

{

cout<<"Elemento ["<<i+1<<"]["<<j+1<<"] de A: ";

cin>>A[i][j];


}

cout<<"Matriz A introducida! "<<endl;


//Ahora vamos a rellenar los vectores

cout<<"Vamos a rellenar el vector de soluciones: "<<endl;

for (int i=0; i<n; i++)

{

    cout<<"Elemento ["<<i+1<<"] de b: ";

    cin>>b[i];

}


//Vamos a mostrar las matrices
```

```
cout<<"La matriz A introducida es: "<<endl;
for (int i=0; i<n; i++)
for (int j=0; j<n; j++)
{
cout<<A[i][j]<<" ";
if (j==n-1)
{cout<<endl;}
}
cout<<endl<<"Y el espacio b de soluciones es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<b[i]<<" "<<endl;
}
jacobi(A, b, x, n);
gaussseidel (A, b, x, n);
LU (A,b,x,n);
cout<<endl<<endl<<endl<<"Terminado!"<<endl;

//Boramos la memoria dinámica

delete [] b;
delete [] x;
for (int i=0; i<n; i++)
{
delete[] A[i];
}

//Líneas comunes en cada programa con dev c++
system ("PAUSE");
return 0;
}



//Cuerpo de las funciones
```

```
float pe(double a[], double b[], int n)
{
 float acu =0;

 for (int i = 0; i < n; i++)
 acu += a[i] * b[i];
 return acu;
}



float norma( double a[], int n)
{
 float aux;

 aux = pe(a, a, n);
 aux = sqrt(aux);
 return aux;
}



void jacobi(double **A, double *b, double *x, int n)
{
     double oldx[n];
     int c=0;

     //Vamos a hacer una primera aproximación de x.
     for(int i=0; i<n; i++)
     {
             x[i]=b[i]/A[i][i];
     }
     do
     {
         for  (int i=0; i<n; i++)
         {
```

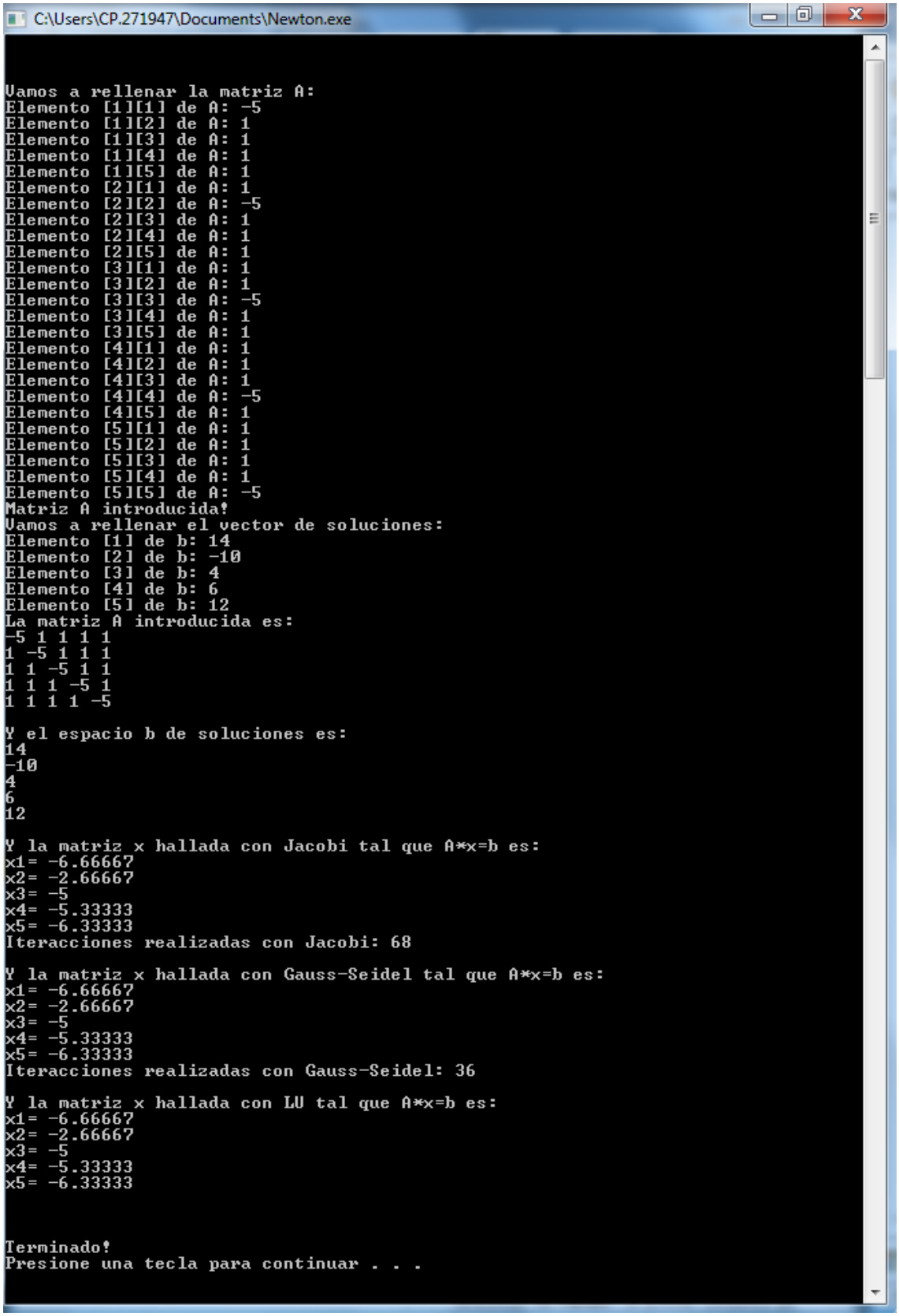
```
            oldx[i]=x[i];
        }
        for (int i=0; i<n; i++)
        {
            x[i]=b[i]/A[i][i];
            for (int j=0; j<n; j++)
            {
                if (j!=i)
                {
                x[i]=x[i]-(A[i][j]/A[i][i])*oldx[j];
                }
            }
        }
        c=c+1;
    }while ((fabs(norma(x, n)-norma(oldx,n)))>tolerancia);
    cout<<endl<<"Y la matriz x hallada con Jacobi tal que A*x=b es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<"x"<<i+1<<"= "<<x[i]<<endl;
}
    cout<<"Iteracciones realizadas con Jacobi: "<<c<<endl;
}

void gaussseidel (double **A, double *b, double *x, int n)
{
double oldx[n];
int c=0;

    //Vamos a hacer una primera aproximación de x.
    for(int i=0; i<n; i++)
    {
            x[i]=b[i]/A[i][i];
    }
    do
```

```
     {
          for  (int i=0; i<n; i++)
          {
               oldx[i]=x[i];
          }
          for (int i=0; i<n; i++)
          {
              x[i]=b[i]/A[i][i];
              for (int j=0; j<n; j++)
              {
                  if (j!=i)
                  {
                  x[i]=x[i]-(A[i][j]/A[i][i])*x[j];
                  }
              }
          }
          c=c+1;
     }while ((fabs(norma(x, n)-norma(oldx,n)))>tolerancia);
     cout<<endl<<"Y la matriz x hallada con Gauss-Seidel tal que A*x=b es: "<<endl;
for (int i=0; i<n; i++)
{
cout<<"x"<<i+1<<"= "<<x[i]<<endl;
}
cout<<"Iteracciones realizadas con Gauss-Seidel: "<<c;


}


void LU(double **A, double *b, double *x, int n)
{
//Tenemos primero que definir dos matrices, la L y la U:
double L[n][n], U[n][n], z[n];
double sumU, sumL, sumz,sumx;
//Ahora que ya están definidas tenemos que darles valores, para eso utilizaremos un loop.
//Loops utilizados para hallar LU
```

```
for(int i=0; i<=n-1;i++)

for(int j=0; j<=n-1;j++)

{

//Caso general de L y U if (i<=j) //Condicional para hacer U

       {

         if (i<j) //Condicional del caso triángulo superior de L

            {L[i][j]=0;}

         if (i==j) //Condicional del caso diagonal de L

            {L[i][j]=1;}

         sumU=0;

         for(int k=0; k<=i-1; k++) //Loop con respecto a sumU

         {sumU=sumU+L[i][k]*U[k][j];}

         U[i][j]=A[i][j]-sumU;

         }

         if (i>j) //Condicional para hacer L

         {

         U[i][j]=0;

         sumL=0;

         for(int k=0; k<=j-1; k++) //Loop con respecto a sumL

         {sumL=sumL+L[i][k]*U[k][j];}

         L[i][j]=(A[i][j]-sumL)/U[j][j];

         }

}

//Loops utilizados para hallar z

for(int i=0; i<=n-1;i++) //Loop principal

{

        sumz=0;

        for(int j=0; j<=i-1;j++) //Loop utilizado para hallar el sumatorio de z

                {

                  sumz=sumz+L[i][j]*z[j];

                }

        z[i]=b[i]-sumz;

}

//Loops utilzados para hallar x
```

```
for(int i=n-1; i>=0; i--) //Loop principal

{

  double cx=0, sumx=0;

  for (int j=i+1; j<=n-1; j++) //Loop utilizado para hallar el sumatoriot

  {

  sumx=sumx+U[i][j]*x[j];

  }

  x[i]=(z[i]-sumx)/U[i][i];

}

cout<<endl<<endl<<"Y la matriz x hallada con LU tal que A*x=b es: "<<endl;

for (int i=0; i<n; i++)

{

cout<<"x"<<i+1<<"= "<<x[i]<<endl;

}

}
```

RESULTADO B


C:\Users\CP.271947\Documents\Newton.exe
Vamos a rellenar la matriz A:
Elemento [1][1] de A: -5
Elemento [1][2] de A: 1
Elemento [1][3] de A: 1
Elemento [1][4] de A: 1
Elemento [1][5] de A: 1
Elemento [2][1] de A: 1
Elemento [2][2] de A: -5
Elemento [2][3] de A: 1
Elemento [2][4] de A: 1
Elemento [2][5] de A: 1
Elemento [3][1] de A: 1
Elemento [3][2] de A: 1
Elemento [3][3] de A: -5
Elemento [3][4] de A: 1
Elemento [3][5] de A: 1
Elemento [4][1] de A: 1
Elemento [4][2] de A: 1
Elemento [4][3] de A: 1
Elemento [4][4] de A: -5
Elemento [4][5] de A: 1
Elemento [5][1] de A: 1
Elemento [5][2] de A: 1
Elemento [5][3] de A: 1
Elemento [5][4] de A: 1
Elemento [5][5] de A: -5
Matriz A introducida!
Vamos a rellenar el vector de soluciones:
Elemento [1] de b: 14
Elemento [2] de b: -10
Elemento [3] de b: 4
Elemento [4] de b: 6
Elemento [5] de b: 12
La matriz A introducida es:
-5 1 1 1 1
1 -5 1 1 1
1 1 -5 1 1
1 1 1 -5 1
1 1 1 1 -5
Y el espacio b de soluciones es:
14
-10
4
6
12
Y la matriz x hallada con Jacobi tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Iteracciones realizadas con Jacobi: 68
Y la matriz x hallada con Gauss-Seidel tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Iteracciones realizadas con Gauss-Seidel: 36
Y la matriz x hallada con LU tal que A*x=b es:
x1= -6.66667
x2= -2.66667
x3= -5
x4= -5.33333
x5= -6.33333
Terminado!
Presione una tecla para continuar . . .

COMENTARIOS

- He realizado ambas prácticas con éxito, creo. Ambos métodos son satisfactorios.
- Vemos que el método de Gauss-Seidel da menos iteraciones, lo que resulta más ventajoso a lo hora de computar.
- Vemos que el sistema propuesto es dominante en la diagonal, por lo que el método converge.
- Siento indiferencia por el método que usar, aunque me resulta más interesante el de LU.
- He realizado el ejercicio con punteros.
- Para el flujo de entrada bastaría con cambiar las líneas que definen A y poner estas

```
double A[n][n];
      double b[n]={14,-10,4,6,12}; //resultados
      double x[n];
      double e;
      ifstream fentrada ("matriz.dat");
      ofstream fsalida ("salida.dat");
      while (!fentrada.eof())
      {
            cout<<"Matriz A "<<endl;
            for (int i=0;i<n;i++)
            {
                  for (int j=0; j<n; j++)
                  {
                        fentrada>>e;
                        A[i][j]=e;
                        cout<<A[i][j]<<" ";
                  }
                  cout<<endl;
            }
            fentrada.close();
            fsalida.close();
      }
```

Y crear un fichero en el bloc de notas guardado como .dat y con esta información -5,1,1,1,1,1,-5,1,1,1,1,1,-5,1,1,1,1,1,-5,1,1,1,1,1,-5