Práctica 2
Resolución de sistemas de ecuaciones lineales (SEL)
Ax=b x=LinearSolve[A,b]

Métodos Directos

■ Método de Gauss y Gauss−Jordan

■ Sistema triangular superior, Método de sustituciones regresivas

```
In[390]:= U = {{1, 2, -1, 3, -4}, {0, 6, 0, 0, 1}, {0, 0, 2, 2, 0}, {0, 0, 0, 1, -4}, {0, 0, 0, 0, 5}};
          MatrixForm[U]
          b = {25, 24, -15, 40, 61};
```

Out[391]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -1 & 3 & -4 \\ 0 & 6 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

```
In[393]:= n = Length[U];
          x = b;
          For[i = n, i ≥ 1, i--,
           x[[i]] = (b[[i]] - Sum[U[[i, k]] * x[[k]], {k, i + 1, n}]) / U[[i, i]]]
          x
          LinearSolve[U, b]
          U.x == b
```

Out[396]= $\left\{-\frac{1757}{6}, \frac{59}{30}, -\frac{963}{10}, \frac{444}{5}, \frac{61}{5}\right\}$

Out[397]= $\left\{-\frac{1757}{6}, \frac{59}{30}, -\frac{963}{10}, \frac{444}{5}, \frac{61}{5}\right\}$

Out[398]= True

■ Método de Gauss

```
In[399]:= A = {{1, 2, 3, 4}, {2, 20, 18, 16}, {3, 18, 19, 21}, {4, 16, 21, 33}};
          A = N[A];
          b = {4, 6, 8, 2};
```

```
In[402]:= n = Length[A];
          For[i = 1, i ≤ n, i++, For[j = i + 1, j ≤ n, j++, pivote = A[[j, i]] / A[[i, i]];
            A[[j]] = A[[j]] - pivote * A[[i]];
            b[[j]] = b[[j]] - pivote * b[[i]]]]
          MatrixForm[A]
          MatrixForm[b]
```

Out[404]//MatrixForm=

$$\begin{pmatrix} 1. & 2. & 3. & 4. \\ 0. & 16. & 12. & 8. \\ 0. & 0. & 1. & 3. \\ 0. & 0. & 0. & 4. \end{pmatrix}$$

Out[405]//MatrixForm=

$$\begin{pmatrix} 4 \\ -2. \\ -2.5 \\ -5.5 \end{pmatrix}$$

■ Método de Gauss–Jordan

```
In[406]:= A = {{1, 2, 3, 4}, {2, 20, 18, 16}, {3, 18, 19, 21}, {4, 16, 21, 33}};
          A = N[A];
          b = {4, 6, 8, 2};
          LinearSolve[A, b]
```

```
Out[409]= {5.937500000000001, -0.6562499999999999, 1.6249999999999998, -1.375}
```

```
In[410]:= n = Length[A];
          For[i = 1, i ≤ n, i++, For[j = i + 1, j ≤ n, j++, pivote = -A[[j, i]] / A[[i, i]];
            A[[j]] = A[[j]] + pivote * A[[i]];
            b[[j]] = b[[j]] + pivote * b[[i]]];
           For[j = 1, j ≤ i - 1, j++, pivote = -A[[j, i]] / A[[i, i]];
            A[[j]] = A[[j]] + pivote * A[[i]];
            b[[j]] = b[[j]] + pivote * b[[i]]]
          ]
          MatrixForm[A]
```

Out[412]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & 0. \\ 0. & 16. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 4. \end{pmatrix}$$

```
In[413]:= x = b;
          For[i = 1, i ≤ n, i++, x[[i]] = b[[i]] / A[[i, i]]];
          x

Out[415]= {5.9375, -0.65625, 1.625, -1.375}
```

■ Condicionamiento

```
In[416]:= A = {{10, 7, 8, 7}, {7, 5, 6, 5}, {8, 6, 10, 9}, {7, 5, 9, 10}};
          A = N[A];
          radioespectraldeA = Max[Abs[Eigenvalues[A]]]
          radioespectraldesuinversa = Max[Abs[Eigenvalues[Inverse[A]]]]
          condicionamientooptimo = radioespectraldeA * radioespectraldesuinversa

Out[418]= 30.288685345802147

Out[419]= 98.52169771010375

Out[420]= 2984.0927016755686
```

■ Métodos de descomposición directa

■ Descomposición LU

```
In[421]:= A = {{1, 2, 4}, {2, 9, 7}, {-1, 8, -2}};
          L = {{1, 0, 0}, {2, 1, 0}, {-1, 2, 4}};
          U = {{1, 2, 4}, {0, 5, -1}, {0, 0, 1}};
          L.U == A

Out[424]= True
```

```
In[425]:= b = {10, 9, -2};
          y = LinearSolve[L, b];
          x = LinearSolve[U, y];
          A.x == b
```

```
Out[428]= True
```

■ Descomposición LU de Doolittle

```
In[429]:= n = Dimensions[A][[1]];
          U = 0 * A;
          L = IdentityMatrix[n];
          For[k = 1, k <= n, k++,
            U[[k, k]] = A[[k, k]] - Sum[L[[k, t]] * U[[t, k]], {t, 1, k - 1}];
            For[j = k + 1, j <= n, j++,
             U[[k, j]] = A[[k, j]] - Sum[L[[k, t]] * U[[t, j]], {t, 1, k - 1}]];
              For[i = k + 1, i <= n, i++,
             L[[i, k]] = (A[[i, k]] - Sum[L[[i, t]] * U[[t, k]], {t, 1, k - 1}]) / U[[k, k]]]
             ];
          MatrixForm[L]
          MatrixForm[U]
          A == L.U
```

```
Out[433]//MatrixForm=
⎛ 1   0  0 ⎞
⎜ 2   1  0 ⎟
⎝ -1  2  1 ⎠
```

```
Out[434]//MatrixForm=
⎛ 1  2   4 ⎞
⎜ 0  5  -1 ⎟
⎝ 0  0   4 ⎠
```

```
Out[435]= True
```

■ Sobre el comando LUDecomposition

```
In[436]:= LUDecomposition[A]
```

```
Out[436]= {{{1, 2, 4}, {2, 5, -1}, {-1, 2, 4}}, {1, 2, 3}, 1}
```

```
In[437]:= desc = LUDecomposition[A][[1]];
          auxL = {{0, 0, 0}, {1, 0, 0}, {1, 1, 0}};
          auxU = {{1, 1, 1}, {0, 1, 1}, {0, 0, 1}};
          U = desc * auxU;
          L = desc * auxL + IdentityMatrix[n];
          A == L.U
```

```
Out[442]= True
```

■ Descomposición LL^t de Cholesky

```
In[443]:= A = {{16, -4, 0, 4, 0, -4}, {-4, 2, 2, 0, 0, 3}, {0, 2, 13, 5, 0, 13},
             {4, 0, 5, 4, -3, 0}, {0, 0, 0, -3, 45, 12}, {-4, 3, 13, 0, 12, 31}};
          A = A // N;
          A == Transpose[A]  (* verificación de ser simétrica *)
          Min[Eigenvalues[A]] > 0  (*verificación de ser definida positiva,
            sii todos los valores son propios reales positivos*)
```

```
Out[445]= True
```

```
Out[446]= True
```

```
In[447]:= n = Length[A];
          L = 0 * A;
          For[k = 1, k ≤ n, k++, L[[k, k]] = Sqrt[A[[k, k]] - Sum[L[[k, j]]^2, {j, 1, k - 1}]];
           For[i = k + 1, i ≤ n, i++,
            L[[i, k]] = (A[[i, k]] - Sum[L[[i, j]] * L[[k, j]], {j, 1, k - 1}]) / L[[k, k]]]]
          MatrixForm[L]
          A == L.Transpose[L]
```

```
Out[450]//MatrixForm=
⎛ 4.   0   0   0   0   0 ⎞
⎜-1.  1.   0   0   0   0 ⎟
⎜ 0.   2.  3.   0   0   0 ⎟
⎜ 1.   1.  1.   1.   0   0 ⎟
⎜ 0.   0.  0.  -3.  6.   0 ⎟
⎝-1.   2.  3.  -4.  0.  1. ⎠
```

```
Out[451]= True
```

■ Descomposición QR

```
In[452]:= A = {{1, 2, 4}, {2, 9, 7}, {-1, 8, -2}};
          {Q, R} = QRDecomposition[A];
          Q = Transpose[Q];
          A == Q.R
```

```
Out[455]= True
```

Métodos Iterativos

■ Descomposición previa A=M−L−U

```
In[456]:= A = {{2, 1, -1}, {-2, 4, 3}, {6, 5, 9}};
          A = N[A];
          b = {14 / 3, 67, 24};
```

```
In[459]:= n = Length[A];
          auxL = Table[If[i > j, 1, 0], {i, n}, {j, n}];
          auxU = Table[If[i < j, 1, 0], {i, n}, {j, n}];
          L = (-A) * auxL;          (* OJO, NO ES UN PRODUCTO MATRICIAL *)
          U = (-A) * auxU;
          d = A + L + U;
          A == d - L - U
          Print[MatrixForm[A], " = ", MatrixForm[d],
           " - ", MatrixForm[L], " - ", MatrixForm[U]]
```

```
Out[465]= True
```

$$\begin{pmatrix} 2. & 1. & -1. \\ -2. & 4. & 3. \\ 6. & 5. & 9. \end{pmatrix} = \begin{pmatrix} 2. & 0. & 0. \\ 0. & 4. & 0. \\ 0. & 0. & 9. \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 2. & 0 & 0 \\ -6. & -5. & 0 \end{pmatrix} - \begin{pmatrix} 0 & -1. & 1. \\ 0 & 0 & -3. \\ 0 & 0 & 0 \end{pmatrix}$$

■ Jacobi

```
In[467]:= A = N[A];
          n = Length[A];
          tolerancia = 10^(-5);
          error = 10 * tolerancia;
          maxiter = 100;
          paso = 0;
          x = Table[0, {n}];
          y = x;
          While[paso < maxiter && error > tolerancia,
           Do[y[[i]] =
              (b[[i]] - Sum[A[[i, j]] * x[[j]], {j, 1, i - 1}] - Sum[A[[i, j]] * x[[j]], {j, i + 1, n}]) /
               A[[i, i]], {i, n}];
           error = Sqrt[(x - y).(x - y)];
           x = y;
           paso = paso + 1
          ]
          Print["Solución aproximada: ", x];
          Print["Iteraciones realizadas: ", paso, ",      Tope de iteraciones: ", maxiter];
          Print["Error permitido: ", tolerancia, ",        Error cometido: ", error];
          Print["Residuo (= |Ax-b|): ", Sqrt[(A.x - b).(A.x - b)]];
```

```
Solución aproximada: {-5.999999258232634, 14.9999951697424, -1.6666618742971913}

Iteraciones realizadas: 55,      Tope de iteraciones: 100

Error permitido: 1/100000,       Error cometido: 9.945790631382909×10^-6

Residuo (= |Ax-b|): 0.00002562327035406438
```

■ Convergencia, estudio matricial

```
In[480]:= B = Inverse[d].(L + U);
          Max[Abs[Eigenvalues[B]]]
```

```
Out[481]= 0.7710573095414456
```

■ Gauss-Seidel

```
In[482]:= A = N[A];
          n = Length[A];
          iteraciones = 50;
          x = Table[0, {n}];
          For[paso = 0, paso ≤ iteraciones, paso++,
           Do[x[[i]] =
              (b[[i]] - Sum[A[[i, j]] * x[[j]], {j, 1, i - 1}] - Sum[A[[i, j]] * x[[j]], {j, i + 1, n}]) /
               A[[i, i]], {i, n}]
          ]
          Print["Solución aproximada: ", x];
          Print["Iteraciones realizadas: ", iteraciones];
          Print["Residuo (= |Ax-b|): ", Sqrt[(A.x - b).(A.x - b)]];
```

```
Solución aproximada: {-5.999999999999993, 15.000000000000002, -1.666666666666673}

Iteraciones realizadas: 50

Residuo (= |Ax-b|): 2.7303361614164952×10^-14
```

■ Convergencia, estudio matricial

```
In[490]:= B = Inverse[d - L].U;
          Max[Abs[Eigenvalues[B]]]

Out[491]= 0.49999999999999994
```

■ Relajación

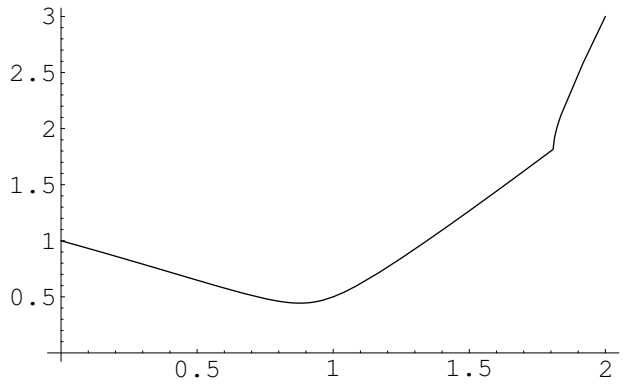
```
In[492]:= A = N[A];
          w = 0.8;
          n = Length[A];
          iteraciones = 50;
          x = Table[0, {n}];
          For[paso = 0, paso ≤ iteraciones, paso++,
           Do[x[[i]] = w * (b[[i]] - Sum[A[[i, j]] * x[[j]], {j, 1, i - 1}] -
                  Sum[A[[i, j]] * x[[j]], {j, i + 1, n}]) / A[[i, i]] + (1 - w) * x[[i]], {i, n}]]
          Print["Solución aproximada: ", x];
          Print["Iteraciones realizadas: ", iteraciones];
          Print["Residuo (= |Ax-b|): ", Sqrt[(A.x - b).(A.x - b)]];
```

```
Solución aproximada: {-6., 15., -1.6666666666666665}

Iteraciones realizadas: 50

Residuo (= |Ax-b|): 8.881784197001252×10^-16
```

■ Convergencia,estudio matricial

```
In[501]:= B = Inverse[d / w - L].((1 - w) * d / w + U);
          Max[Abs[Eigenvalues[B]]]

Out[502]= 0.46061710339370293
```

```
In[503]:= f[w_] := Max[Abs[Eigenvalues[Inverse[d / w - L].((1 - w) * d / w + U)]]]
          Plot[f[w], {w, 0, 2}];
```



■ Ejercicios

1.–Programa la resolución de un sistema triangular inferior.
2.–Calcula el condicionamiento de una matriz usando la norma 1.
3.–Programa la descomposición LU de una matriz con la elección de Crout.
4.–Reescribe el programa para que detecte si una matriz NO puede ser descompuesta LU.
5.–Reescribe el programa de Cholesky para que detecte si la matriz no admite descomposición.
6.–Modifica el programa de Gauss–Seidelpara que se detenga cuando la distancia eucl´idea entre 2 iteraciones sea inferior a 10^(–5).
7.–¿Cómo detectar´ias numéricamente que un método no converge?
8.–Repite el ejercicio 6 para Relajación.