Práctica: Métodos de resolución de ecuaciones lineales.

- **Objetivo:** Aplicar dos técnicas de resolución de sistemas de ecuaciones lineales:
  - Un método finito basado en la descomposición LU de la matriz de coeficientes A
  - El método iterativo de Gauss-Seidel.
  - El método iterativo de Jacobi.
  - Método finito basado en la descomposición QR de la matriz de coeficientes A (ampliación)

Matemáticas para la Computación

Emiliano Torres

Método de descomposición LU

- La **descomposición LU** de una matriz cuadrada *A* es aquella que escribe *A* como el producto de dos matrices triangulares, L y U, tales que L es **triangular inferior** y U es **triangular superior**.
- La descomposición LU se puede utilizar para resolver sistemas de ecuaciones **Ax=b**. Si **A=LU**, se puede reescribir la expresión anterior como **LUx=b**. Si llamamos **z** a la matriz de n filas resultado del producto de las matrices Ux, se tiene **Lz=b.**
- Se plantea un algoritmo para resolver el sistema de ecuaciones empleando dos etapas:
  - 1º obtenemos z aplicando el **algoritmo de sustitución progresiva** (1) a **Lz=b.**

$$(1)\, z_i = (b_i - \sum_{j=1}^{i-1} l_{ij} z_j) / l_{ii} \quad (i = 1,2,\ldots,n)$$

  - 2º obtenemos x aplicando el **algoritmo de sustitución regresiva** (2) a **Ux=z**

$$(2)\, x_i = (z_i - \sum_{j=i+1}^{n} u_{ij} x_j) / u_{ii} \quad (i = n, n-1,\ldots,1)$$

Algoritmo de factorización LU

- Input n, A
- For k=1,2,...,n do
  - Especificar un valor para l_{kk} o u_{kk} (l_{kk}=1 factorización de Doolittle, u_{kk}=1 factorización de Crout)
  - Calcular el otro término mediante: $l_{kk}u_{kk} = a_{kk} - \sum_{s=1}^{k-1} l_{ks}u_{sk}$
  - For j=k+1, k+2, ..., n do

    $$u_{kj} = (a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sk}) / l_{kk}$$
  - End
  - For i=k+1, k+2, ..., n do

    $$l_{ik} = (a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk}) / u_{kk}$$
  - End
- End
- Output L, U

Práctica

1. Método de **descomposición LU**:

- Emplear la función intrínseca de MatLab **[l,u,p]=lu(A)** para obtener la factorización LU de la matriz de coeficientes.
- Programar el algoritmo de sustitución progresiva mediante una función **z=suspro(l,b)**
- Programar el algoritmo de sustitución regresiva mediante la función **x=susreg(u,z)**
- Escribir una macro que utilice las funciones anteriores para resolver los sistemas de ecuaciones de la práctica.

Método de Gauss-Seidel

- Es un método iterativo que resulta ser un método bastante eficiente.

$$
\begin{array}{ccccc}
a_{11}x_1 & +a_{12}x_2 & +\cdots & +a_{1n}x_n & = & b_1 \\
a_{21}x_1 & +a_{22}x_2 & +\cdots & +a_{2n}x_n & = & b_2 \\
 & \vdots & & & & \vdots \\
a_{n1}x_1 & +a_{n2}x_2 & +\cdots & +a_{nn}x_n & = & b_n
\end{array}
$$

- De la ecuación 1 despejamos x_1, de la ecuación 2 despejemos x_2, …, de la ecuación *n* despejemos x_n . Esto nos da el siguiente conjunto de ecuaciones:

$$
\begin{aligned}
x_1 &= \frac{b_1 - a_{22}x_2 - \cdots - a_{1n}x_n}{a_{11}} \\
x_2 &= \frac{b_2 - a_{21}x_1 - \cdots - a_{2n}x_n}{a_{22}} \\
&\vdots \\
x_n &= \frac{b_n - a_{11}x_1 - \cdots - a_{nn-1}x_{n-1}}{a_{nn}}
\end{aligned}
$$

Método de Gauss-Seidel

- El conjunto de ecuaciones forman las fórmulas iterativas. Para comenzar el proceso iterativo, se da el valor de cero a las variables $x_{2,\ldots,}\ x_n$; esto da un primer valor para x_1.

$$x_1 = \frac{b_1}{a_{11}}$$

- Se sustituye el valor de x_1 en la ecuación 2, y las variables siguen teniendo el valor de cero. Esto da el siguiente valor para x_2:

$$x_2 = \frac{b_2 - a_{21}\left(\frac{b_1}{a_{11}}\right)}{a_{22}}$$

- Estos valores de x_1 y x_2, los sustituímos en la ecuación 3, mientras que $x_{3,\ldots,}\ x_n$ siguen teniendo el valor de cero; y así sucesivamente hasta llegar a la última ecuación. Con este primer paso del proceso iterativo se obtiene la primera lista de valores para las incógnitas:

$$\begin{aligned} x_1 &= \alpha_1 \\ x_2 &= \alpha_2 \\ &\vdots \\ x_n &= \alpha_n \end{aligned}$$

Método de Gauss-Seidel

- Se vuelve a repetir el proceso sustituyendo estos últimos datos en vez de ceros como al inicio, y se obtiene una segunda lista de valores para cada una de las incógnitas:

$$\begin{aligned} x_1 &= \beta_1 \\ x_2 &= \beta_2 \\ &\vdots \\ x_n &= \beta_n \end{aligned}$$

- Ahora se puede calcular los errores aproximados relativos, respecto a cada una de las incógnitas:

$$\begin{aligned} \left|\in_{a,1}\right| &= \left|\frac{\beta_1 - \alpha_1}{\beta_1} \times 100\%\right| \\ \left|\in_{a,2}\right| &= \left|\frac{\beta_2 - \alpha_2}{\beta_2} \times 100\%\right| \\ &\vdots \\ \left|\in_{a,n}\right| &= \left|\frac{\beta_n - \alpha_n}{\beta_n} \times 100\%\right| \end{aligned}$$

- El proceso se vuelve a repetir hasta que el error sea menor que una cota prefijada.

$$\left|\in_{a,i}\right| < \in_s, \qquad \forall i = 1, 2, \cdots, n$$

Ejemplo del método de Gauss-Seidel

- Para aproximar la solución del sistema hasta que el error sea menor del 1% primero Despejamos las incógnitas:

$$
\begin{array}{rrrcr}
3x_1 & -0.2x_2 & -0.5x_3 & = & 8 \\
0.1x_1 & +7x_2 & +0.4x_3 & = & -19.5 \\
0.4x_1 & -0.1x_2 & +10x_3 & = & 72.4
\end{array}
$$

$$
\begin{aligned}
x_1 &= \frac{8+0.2x_2+0.5x_3}{3} \\
x_2 &= \frac{-19.5-0.1x_1-0.4x_3}{7} \\
x_3 &= \frac{72.4-0.4x_1+0.1x_2}{10}
\end{aligned}
$$

- Despejamos las incógnitas:
- Comienza el proceso iterativo, sustituyendo los valores de $x_2=x_3=0$ en la primera ecuación, para calcular el primer valor de $x_1=2,66667$
- Se sustituye $x_1=2,66667$ y $x_3=0$ en la segunda ecuación, para obtener $x_2= -2.82381$
- Se sustituye $x_1=2,66667$ y $x_2= -2.82381$ en la tercera ecuación, para obtener $x_3= 7.1051$

Ejemplo del método de Gauss-Seidel

- Puesto que todavía no podemos calcular ningún error aproximado, repetimos el proceso pero ahora con los últimos datos obtenidos para las incógnitas.
- Sustituyendo x_2= -2.82381 y x_3= 7.1051 en la ecuación 1 se obtiene x_1= 3.6626. Sustituyendo x_1= 3.6626 y x_3= 7.1051 en la ecuación 2 se obtiene x_2= -3.24404; finalmemente, sustituyendo x_1= 3.6626 y x_2= -3.24404 en la ecuación 3 se obtiene x_3= 7.06106. Así, tenemos la segunda lista de valores de aproximación a la solución del sistema:

$$
\begin{aligned}
x_1 &= 3.6626 \\
x_2 &= -3.24404 \\
x_3 &= 7.06106
\end{aligned}
$$

- Ahora si se puede calcular los errores absolutos para cada una de las incógnitas:

$$
\begin{aligned}
\left|\epsilon_{a,1}\right| &= \left|\frac{3.6626 - 2.66667}{3.6626} \times 100\%\right| = 27\% \\
\left|\epsilon_{a,2}\right| &= \left|\frac{-3.24404 + 2.82381}{-3.24404} \times 100\%\right| = 12.9\% \\
\left|\epsilon_{a,3}\right| &= \left|\frac{7.06106 - 7.1051}{7.06106} \times 100\%\right| = 0.6\%
\end{aligned}
$$

Ejemplo del método de Gauss-Seidel

- Puesto que no se ha logrado el objetivo, debemos repetir el mismo proceso con los últimos valores obtenidos de cada una de las incógnitas. Nótese que aunque el error aproximado de x_3 ya cumple con ser menor al 1%, esto se debe de cumplir para los tres errores aproximados!
- Por lo tanto repitiendo el mismo proceso se obtiene:

$$x_1 = 3.62724$$
$$x_2 = -3.24102$$
$$x_3 = 7.06250$$

- Y en este caso los errores aproximados son:

$$\left|\epsilon_{a,1}\right| = 0.09\%$$
$$\left|\epsilon_{a,2}\right| = 0.07\%$$
$$\left|\epsilon_{a,3}\right| = 0.001\%$$

- Ahora se ha cumplido el objetivo para cada uno de los errores aproximados. Por lo que la solución aproximada es la obtenida.

Práctica

2. Implementar el **método de Gauss-Seidel** mediante una función de Matlab para resolver un sistema de cuaciones lineales Ax=b generando una serie de vectores. Utilizar la sigiente fórmula de recurrencia:

$$x_i^{(k)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)})/a_{ii}$$

- ***[x,delta,m]=seidel(A,b,Delta,M)***
- Entrada:
  - a : Matriz de coeficientes.
  - b : Matriz columna de términos independientes.
  - Delta : Criterio de convergencia.
  - M : Numero máximo de iteraciones.
- Salida:
  - x : Matriz columna de soluciones.
  - delta : Convergencia alcanzada.
  - m : Número de iteraciones.

Método de Jacobi

- Se escoge una matriz inicial Q que es diagonal y cuyos elementos diagonales son los mismos que los de la matriz A.
- En el método de Gauss-Seidel los valores actualizados de x_i sustituyen de inmediato a los valores anteriores, mientras que en el método de Jacobi todas las componentes nuevas del vector se calculan antes de llevar a cabo la sustitución.
- En el método de Gauss-Seidel los cálculos de x_i dependen de x_1, x_2, ...,x_{i-1}

Método de Jacobi

- Se obtienen todos los xi a partir de la matriz diagonal.

$$x_1 = \frac{b_1}{a_{11}}$$

$$\begin{aligned} x_1 &= \frac{b_1 - a_{22}x_2 - \cdots - a_{1n}x_n}{a_{11}} \\ x_2 &= \frac{b_2 - a_{21}x_1 - \cdots - a_{2n}x_n}{a_{22}} \\ &\vdots \\ x_n &= \frac{b_n - a_{11}x_1 - \cdots - a_{nn-1}x_{n-1}}{a_{nn}} \end{aligned}$$

$$\begin{aligned} x_1 &= \alpha_1 \\ x_2 &= \alpha_2 \\ &\vdots \\ x_n &= \alpha_n \end{aligned}$$

- Se sustituyen los valores anteriores en las ecuaciones se obtiene la siguiente lista de valores para las incógnitas:

$$\begin{aligned} x_1 &= \beta_1 \\ x_2 &= \beta_2 \\ &\vdots \\ x_n &= \beta_n \end{aligned}$$

Método de Jacobi

- Ahora se puede calcular los errores aproximados relativos, respecto a cada una de las incógnitas:

$$|\in_{a,1}| = \left|\frac{\beta_1 - \alpha_1}{\beta_1} \times 100\%\right|$$

$$|\in_{a,2}| = \left|\frac{\beta_2 - \alpha_2}{\beta_2} \times 100\%\right|$$

$$\vdots$$

$$|\in_{a,n}| = \left|\frac{\beta_n - \alpha_n}{\beta_n} \times 100\%\right|$$

- El proceso se vuelve a repetir hasta que el error sea menor que una cota prefijada.

$$|\in_{a,i}| < \in_s, \qquad \forall i = 1,2,\cdots,n$$

Práctica

2. Implementar el **método de Jacobi** mediante una función de Matlab para resolver un sistema de cuaciones lineales Ax=b generando una serie de vectores. Utilizar la sigiente fórmula de recurrencia:

$$x_i^{(k)} = (b_i - \sum_{j=1,\, j \neq I}^{n} a_{ij} x_j^{(k-1)}) / a_{ii}$$

- ***[x,delta,m]=jacobi(A,b,Delta,M)***
- Entrada:
  - a : Matriz de coeficientes.
  - b : Matriz columna de términos independientes.
  - Delta : Criterio de convergencia.
  - M : Numero máximo de iteraciones.
- Salida:
  - x : Matriz columna de soluciones.
  - delta : Convergencia alcanzada.
  - m : Número de iteraciones.

Método de descomposición QR

- La **descomposición *QR*** de una matriz cuadrada *A* es aquella que escribe *A* como el producto de dos matrices cuadradas, *Q* y *R*, tales que *Q* es **ortogonal** y *R* es **triangular superior**.
- La descomposición *QR* se puede utilizar para resolver sistemas de ecuaciones **Ax=b**. Si **A=QR**, se puede reescribir la expresión anterior como **QRx=b**. Si ahora se multiplica por la izquierda por $\mathbf{Q^{-1}}$, se tiene:

$$Q^{-1}QRx = Q^{-1}b$$

y puesto que *Q* es una matriz ortogonal, finalmente se obtiene:

$$Rx = Q^{T}b$$

$$Rx = c$$

- Este sistema es un sistema triangular superior y por lo tanto fácil de resolver (para ello basta con utilizar el algoritmo de sustitución regresiva).

Ejemplo del método de descomposición QR

- Para el sistema de ecuaciones:

$$\begin{array}{lllll} 3x_1 & -0.2x_2 & -0.5x_3 & = & 8 \\ 0.1x_1 & +7x_2 & +0.4x_3 & = & -19.5 \\ 0.4x_1 & -0.1x_2 & +10x_3 & = & 72.4 \end{array}$$

- Se obtiene la descomposición QR de A:

  Q =

0.9907	-0.0314	-0.1325
0.0330	0.9994	0.0104
0.1321	-0.0147	0.9911

  - R =

3.0282	0.0198	0.8388
0	7.0035	0.2689
0	0	9.9817

- Se calcula c = 16.8450
  -20.8000
  70.4955
- Se obtienen las soluciones:
  x = 3.6277 -3.2411 7.0625

Práctica

3. Implementar el método de **descomposición QR** de la siguiente forma:

- Escribe la función ***[Q, R]=dqr(A)*** *que obtiene la decomposición QR de una matriz A.*
- Escribe la función ***x=susreg(R, c)*** *que implemente el algoritmo de sustitución regresiva.*
  - Entrada:
    - R : Matriz de coeficientes triangular superior.
    - c : Vector de términos independientes.
  - Salida:
    - x : Matriz columna de soluciones.
- *Escribir una macro que utilice las funciones anteriores para resolver los sistemas de ecuaciones de la práctica.*