Actividad N°3: Resolución numérica de SEL

Métodos directos:

1. Considere el sistema de ecuaciones lineales $\mathbf{Ax} = \mathbf{b}$, siendo $\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 12 & 11 & 5 \\ -2 & 9 & 0 \end{bmatrix}$ y $\mathbf{b} = \begin{bmatrix} 1 \\ 17 \\ 18 \end{bmatrix}$.
   (a) Resuelva a mano con el método de Gauss con pivoteo parcial escalado.
   (b) Compruebe sus resultados utilizando la función `Gauss.m`.
   (c) Repita lo anterior utilizando la función `DescompLU.m`. ¿Obtiene diferencias?

2. (a) Desarrolle una función en MATLAB/Octave/Scilab que tome como argumento una matriz cuadrada triangular superior $\mathbf{A}$, el vector lado derecho $\mathbf{b}$, y devuelva una solución $\mathbf{x}$ mediante una sustitución regresiva *(backward sustitutition). Sugerencia:* Ayúdese con la función `DescompLU.m`.
   (b) Repita lo anterior para una sustitución progresiva *(forward sustitutition).*

3. Se sabe que la matriz de coeficientes del SEL $\mathbf{Ax} = \mathbf{b}$ admite ser factorizada con matrices $\mathbf{L}$, $\mathbf{U}$ y $\mathbf{P}$ de la forma

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 3 & -1 \\ 0 & 0 & -2 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

   Reconstruya la matriz de coeficientes $\mathbf{A}$ y halle la solución del SEL siendo $\mathbf{b} = \begin{bmatrix} -0.2 \\ 2.5 \\ 1.9 \end{bmatrix}$.
   (a) Por el método de Gauss con pivoteo parcial escalado.
   (b) Por factorización incompleta *LU*.
   (c) Calcule la solución exacta usando el comando `mldivide(A,b)` de MATLAB/Octave *(división a izquierda, también conocido como* barra invertida (backslash) `A\b`*)*, o `linsolve(A,b)` de Scilab. Determine si hubo error de redondeo en (*a*) y (*b*).

4. *Aplicación de la factorización LU: inversión de matrices.*
   Considere el sistema de ecuaciones lineales dado en su forma ampliada

$$\left[\begin{array}{rrrr|c} 16 & -120 & 240 & -140 & \\ -120 & 1200 & -2700 & 1680 & \mathbf{e_i} \\ 240 & -2700 & 6480 & -4200 & \\ -140 & 1680 & -4200 & 2800 & \end{array}\right]$$

   donde $\mathbf{e_i}$ es el versor de la base canónica en $\mathbb{R}^4$.
   (a) Obtenga la factorización LU de la matriz de coeficientes $\mathbf{A}$. Explique cuál es la ventaja de usar esta descomposición factorial al resolver sistemas lineales.
   (b) Usando la descomposición anterior, resuelva con algoritmos específicos de *sustitución progresiva* y *regresiva* los sistemas de ecuaciones lineales $\mathbf{Ax_i} = \mathbf{e_i}$ para $\mathbf{i} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}$.
   (c) ¿Qué matriz representa $\mathbf{B} = [\mathbf{x_1}\ \mathbf{x_2}\ \mathbf{x_3}\ \mathbf{x_4}]$ formada por los vectores columnas soluciones de los sistemas lineales anteriores? Compruebe.

Métodos iterativos:

5. Considere el SEL

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

   (a) Calcule a mano las dos primeras iteraciones del *método de Jacobi.*

   (b) Resuelva utilizando la implementación `Jacobi.m` tomando como aproximación inicial $\mathbf{x}^{(0)} = \mathbf{0}$ y una tolerancia de 1×10^{-3}.

   (c) ¿Puede asegurar la convergencia del método de Jacobi en este caso? Justifique.

   (d) Repita los 3 items anteriores utilizando el *método de Gauss-Seidel.*

6. Considere los siguientes SEL

(a) $\begin{cases} -x + 3y = 1 \\ 6x - 2y = 2 \end{cases}$ (b) $\begin{cases} x + z = 2 \\ -x + y = 0 \\ x + 2y - 3z = 0 \end{cases}$ (c) $\begin{cases} 5x - y + z = 10 \\ 2x + 8y - z = 11 \\ -x + y + 4z = 3 \end{cases}$

   (a) Modifique las implementaciones de `Jacobi.m` y `GaussSeidel.m` para que se imprima por pantalla la iteración k-ésima de la aproximación.

   (b) Compare la cantidad de iteraciones necesarias en la resolución de los SEL.

   (c) Resuelva utilizando los métodos de `Jacobi` y de `GaussSeidel` para cierta tolerancia *tol* fijada. ¿Observa algún comportamiento general en estos casos?

   (d) ¿Puede asegurar *a priori* la convergencia de estos métodos iterativos?

7. Una parte muy importante del *Método de Diferencias Finitas (MDF) (en inglés, Finite Difference Method)* para aproximar las soluciones de Ecuaciones en Derivadas Parciales, consiste en resolver sistemas de ecuaciones lineales (SEL) en forma eficiente. Estos sistemas se resuelven en general con *métodos iterativos* que dependen del modelo en cuestión. Considere el siguiente SEL dado en su forma matricial obtenido al resolver por el MDF una ecuación de Laplace $\Delta u(x, y) = 0$ sobre un dominio rectangular $R \subset \mathbb{R}^2$ con condiciones de borde de Dirichlet $u(x, y) = f(x, y)$ sobre la frontera ∂R:

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 \\ 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} -100 \\ -20 \\ -20 \\ -180 \\ -180 \\ 100 \end{bmatrix}$$

   (a) Aproxime la solución utilizando los métodos iterativos de *Jacobi* y *Gauss-Seidel* hasta que se alcance una tolerancia de $tol = 1 \times 10^{-5}$. ¿Cuál converge "más rápido"?

   (b) Verifique que los vectores calculados en (a) son aproximaciones de la solución.

   (c) ¿Puede asegurar *a priori* la convergencia de estos métodos iterativos? Justifique.

Vector residual y Número de condición:

8. Para cada una de las siguientes matrices:

   i) $\mathbf{A} = \begin{bmatrix} 4 & 1 \\ 12 & 2 \end{bmatrix}$

   ii) $\mathbf{B} = \begin{bmatrix} 0 & 3 \\ -5 & 4 \end{bmatrix}$

   iii) $\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 2 & -4 \\ 2 & -5 & 1 \end{bmatrix}$

   iv) $\mathbf{D} = \begin{bmatrix} 2 & -2 & 1 \\ -8 & 11 & 5 \\ 4 & -13 & 3 \end{bmatrix}$

   v) $\mathbf{E} = \begin{bmatrix} \texttt{eps} & 1 \\ 0 & 1 \end{bmatrix}$

   vi) $\mathbf{H_5} = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$

   Esta matriz se conoce como *matriz de Hilbert* y se define como $\mathbf{H_n} = [h_{ij}]$:

   $$h_{ij} = \frac{1}{i+j-1}.$$

   En MATLAB/Octave, `hilb(n)`.
   En Scilab, `inv(testmatrix('hilb',n))`.

   (a) Genere un vector de lado derecho $\mathbf{b_i}$ en cada caso usando el comando `ones` de MATLAB/Octave/Scilab y resuelva el SEL

   $$\mathbf{A}_i\mathbf{x} = \mathbf{b}_i$$

   mediante algún método directo o iterativo.

   (b) Calcule el *vector residual*

   $$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$$

   en los SEL anteriores y la norma del mismo $\|\mathbf{r}\|_p$ para $p = 1, 2, \inf$. Compare los resultados anteriores en una tabla.

9. Para las matrices del ejercicio anterior, calcule con ayuda de MATLAB/Octave/Scilab:

   (a) El *número de condición* $\kappa(\mathbf{A}) = \|\mathbf{A}\|_p.\|\mathbf{A}^{-1}\|_p$ utilizando el comando `inv(A)` para calcular la inversa y el comando `norm(A,p)` para la norma matricial con $p = 1, 2, \inf$.

   (b) Compare los resultados usando el comando `cond(A)` de MATLAB/Octave/Scilab. ¿Están bien condicionadas las matrices?

   (c) ¿Observa alguna relación entre el vector residual $\mathbf{r}$ y el número de condición $\kappa(\mathbf{A})$ de las matrices obtenido?

   Nota: Esta definición del número de condición no se utiliza en las implementaciones debido al costo computacional para calcular una inversa matricial.

10. Modifique los archivos `Jacobi.m` y `GaussSeidel.m` de forma que:

    (a) El criterio de parada sea la norma del *vector residual* $\|\mathbf{b} - \mathbf{A}\mathbf{x}^{(\mathbf{k})}\| < tol$.

    (b) La salida por pantalla sea la iteración k-ésima, la aproximación, el vector residual.

    (c) Compare los tiempos computacionales de estas implementaciones modificadas (usando los comandos `tic` y `toc` de MATLAB/Octave/Scilab) con las implementaciones originales para los ejemplos del Ejercicio 8.

    Nota: Este criterio de parada es malo debido a que el producto matricial $\mathbf{A}\mathbf{x}^{(\mathbf{k})}$ es muy costoso de calcular en cada paso.