Práctica 3. Sistemas de Ecuaciones.

Antes de comenzar con este guion de prácticas prepara todos los métodos vistos en clase en funciones independientes para ser usadas.
Todos los scripts y funciones se copiarán en un Word de resultados, atendiendo a la Sección y número del ejercicio en el que nos encontremos.
Para aquellos scripts o funciones que sean llamados desde consola de comandos se copiará, además, la llamada realizada, así como el resultado generado, tanto numérico como figuras.

1. Matlab incorpora funciones para conocer el condicionamiento de un sistema de ecuaciones a través de las funciones intrínsecas ***cond***, ***rcond*** o ***condest.***

Haciendo uso de las tres funciones anteriores, escribe una función llamada ***condicion.m*** que evalúe los valores de los tres números de condición, el determinante de la matriz y su inversa así como la solución del sistema (aplica el operador \). Esta función tendrá como variables de entrada la matriz **A** del sistema de ecuaciones y el vector independiente **b**.

a) Aplícalo al siguiente sistema de ecuaciones:

$$0.832\, x_1 - 0.448\, x_2 = 1$$

$$0.784\, x_1 + 0.421\, x_2 = 0$$

b) Haz lo mismo con el siguiente sistema de ecuaciones:

$$\boldsymbol{A} = [1.985, -1.358;\ 0.953, -0.652];\ \boldsymbol{b} = [2.212; 1.062].$$

¿Qué pasa si se modifica b(2) por 1.061? ¿Por qué?

c) Haz lo mismo con el siguiente sistema de ecuaciones: $\boldsymbol{A} = [1\ \ 2;\ 2\ \ 4]$, $\boldsymbol{b} = [0; 1]$.

2. Crea una función que permita resolver un sistema de ecuaciones triangular inferior por el método de sustituciones progresivas. Llama a la función **sp.m** y diséñala de tal modo que reciba como parámetros de entrada la matriz de coeficientes A y un vector columna que contenga los términos independientes ***b***, y devuelva como salida un vector columna con las soluciones del sistema ***x***. Aplica la función creada para obtener la solución del sistema:

$$x_1 = 1$$

$$0.5x_1 - 2x_2 = -3.5$$

$$-x_1 + 0.5x_2 + 3x_3 = 9$$

3. Matlab suministra una función propia ***lu.m*** para el cálculo de la factorización *LU*, con pivoteo de filas. La sintaxis es, **[L,U,P]=lu(A)**; donde *L* es una matriz triangular inferior, *U* una matriz triangular superior y *P* es la matriz de permutaciones que da cuenta del cambio de orden aplicado a las ecuaciones del sistema, *P*A=LU.* Aplícalo para resolver el siguiente sistema de ecuaciones

$$\begin{aligned} 0.5x_1 + 2x_2 + 2.5x_3 + 3x_4 &= 24 \\ 0.7x_1 + 5.2x_2 - 3x_3 + x_4 &= 6.1 \\ 0.8x_1 - 6x_2 + 3.4x_3 - 2x_4 &= -9 \\ 2.1x_1 + 3.2x_2 - 4.5x_3 + 2.3x_4 &= 4.2 \end{aligned}$$

4. Sean las siguientes sentencias que determinan las matrices estrictamente inferior *L*, estrictamente superior *U* y diagonal *D* de una matriz ***A***:

*L=A-triu(**A**)*

*U=A-tril(**A**)*

*D=diag(diag(**A**))*

En el método de **Jacobi** la expresión matricial para determinar el vector solución x viene dada por:

x=inv(D)(b-(L+U)*x0)* siendo ***b*** el vector independiente y ***x0*** el vector solución inicial.

Construye una función **jacobi1.m** que determine la primera solución x, partiendo de (0,0,0) como aproximación inicial de la solución. Determina la primera solución del sistema de ecuaciones:

$$3x + y + z = 4$$

$$2x + 5y + 1z = -1$$

$$-x + y + 3z = 4$$

5. Escribe una función **jacobi.m**, con estructura ***x=jacobi(A,b,x0,tol)***, de modo que resuelva un sistema de ecuaciones usando el método de **Jacobi**.

a) Aplícalo al sistema del ejercicio anterior hasta conseguir un error inferior (usa el comando matlab ***norm***) a la tolerancia *tol y* partiendo de un valor ***x0*** como aproximación inicial a la solución. Usa el comando ***while*** y la forma matricial del método de **Jacobi**. Toma una tolerancia de 0.001. Las sucesivas soluciones se deben ver en pantalla en forma de tabla, tal que, en la primera columna aparezca la iteración, en las tres siguientes columnas los elementos del vector solución ***x*** correspondientes a esa iteración ***s*** y en la quinta columna el error entre dos soluciones consecutivas $\boldsymbol{x}^{(s)}$ y $\boldsymbol{x}^{(s-1)}$. Recuerda usar ***norm*** para esto último.

b) Si aplicas tu programa ***jacobi.m***, al sistema de ecuaciones:

$$x+2y+3z=4$$
$$4x+5y+6z=-1$$
$$7x+8y+9z=4.$$

partiendo de (0,0,0) como aproximación inicial de la solución verás que **Jacobi** no converge ya que la matriz del sistema es singular. Para detectar estos casos, añade una condición a tu función ***jacobi.m*** para que cuando el determinante de la matriz del sistema sea menor que 3****eps*** nos diga que no hay solución y se corte la ejecución del programa, por ejemplo, con la orden **return**.

c) Aplica ahora tu nuevo programa de Jacobi al siguiente sistema

$$2x-2y = 4$$
$$2x+3y-z=-1$$
$$5x+2z=4$$

Como verás, el método no converge aunque la matriz del sistema tenga determinante no nulo.

d) En los métodos iterativos es posible previamente determinar la convergencia del método estudiando las propiedades de la matriz de convergencia, que relaciona dos soluciones consecutivas. Ello puede hacerse incorporando al programa ***jacobi.m*** las siguientes sentencias:

```
>> J=-inv(D)*(L+U); %J es la matriz de convergencia del método de Jacobi, que relaciona dos soluciones consecutivas

>>autovalores=eig(J); % eig calcula los autovalores de una matriz (lo verás en álgebra)

>>abs_auto=abs(eig(J));

>>radio_espectral=max(abs(eig(J)))

>>if radio_espectral>1

>>        disp('El metodo no converge. No da la solución del sistema')

>>        return

>>end
```

Escribe tu programa final **jacobi.m**, las ejecuciones para los tres sistemas de ecuaciones y los resultados.

6. (HACER EN CASA) Escribir una función **x=jacobi_amor(A,b,x0,w,tol)** que resuelva un sistema de ecuaciones por el método iterativo Jacobi amortiguado, hasta conseguir un error de convergencia inferior a la tolerancia *tol*, partiendo de un valor **x0** como aproximación inicial a la solución y ***w*** como peso.

Nota: Con añadir en tu programa **jacobi.m** la sentencia de amortiguamiento tras la sentencia que define el método de Jacobi es suficiente. Además debe tenerse en cuenta que la matriz de convergencia cambia y ya no es la misma que la del método de Jacobi. ¿Cuál será ahora la matriz de convergencia?

a) Aplica tu función al siguiente sistema de ecuaciones con una tolerancia de 0.001, un peso de 0.9 y vector solución inicial [0;0;0]. Te tiene que salir un radio espectral de 0.9463, por tanto el método converge.

$$2x-2y = 4$$
$$2x+3y-z=-1$$
$$-5x+2z=4$$

b) Si la solución inicial fuera x0=[3; 1; 10], ¿cuántas iteraciones se realizan hasta llegar a la solución? ¿Cuál es tu conclusión?

c) Si el peso valiera 1.1, ¿cuál será el valor del radio espectral para el sistema de ecuaciones anterior?

7. Resolución de sistemas de ecuaciones por **Gauss-Seidel.**

a) Escribe una función de la forma **xs1 = gseidel1(A,b,xs)** que realice una iteración del método de Gauss-Seidel y aplícalo al siguiente sistema de ecuaciones con xs = [0; 0; 0]:

$$2x-2y = 4$$
$$2x+3y-z=-1$$
$$5x+2z=4.$$

b) Escribe una función **gseidel.m** que resuelva de manera iterativa el sistema de ecuaciones anterior por el método de Gauss-Seidel y que llame a la función escrita en el apartado anterior. Considera un valor de tolerancia para la solución de 0.001. No olvides incluir las consideraciones sobre el valor del determinante de A y de la matriz de convergencia.