MÓDULO SE: SISTEMAS DE ECUACIONES

Alumno:

Lee detenidamente los enunciados. Copia las funciones y scripts que crees a lo largo de la practica, así como los resultados de las ejecuciones en este documento .

SE.1. Condicionamiento de una matriz con MATLAB.

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 nos diga los valores de los tres números de condición, y del determinante de la matriz :

$$0.832\ x_1 - 0.448\ x_2 = 1$$
$$0.784\ x_1 + 0.421\ x_2 = 0$$

Considera la matriz A del sistema y el vector independiente b como variables de entrada.

SE.2. Sistemas Fáciles de Resolver

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

$$\begin{aligned} x_1 &= 1 \\ 0.5x_1 - 2x_2 &= -3.5 \\ -x_1 + 0.5x_2 + 3x_3 &= 9 \end{aligned}$$

SE.3. Factorización LU (opcional)

Escribe un programa que permita obtener las matrices correspondientes a la factorización LU, a partir de la matriz de coeficientes de un sistema de ecuaciones lineales. Aplícalo a la resolución del sistema,

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

NOTA: Matlab suministra una función propia 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.

La función en Matlab debe admitir como entradas la matriz de coeficientes de un sistema y un vector columna con los términos independientes, y debe devolver, las matrices L,U y P y un vector columna con las soluciones del sistema.

SE.4. Métodos Iterativos

Sea la siguiente función

```
function [x]=jacobi1(A,b,x0)
%
% x=jacobi1(A,b,x0). Aplica UNA iteración del método iterativo de Jacobi
%                    al sistema de ecuaciones A . x = b, usando x0 como valor
%                    de la aproximación a la solución en la iteración anterior.
% Valores de entrada: A (matriz con los coeficientes del sistema)
%       b  (vector COLUMNA con los términos independientes del sistema)
%       x0 (vector COLUMNA con la aproximación a la solución en la iteración anterior)
% Valores de salida: x (nueva aproximación a la solución)
%
L=A-triu(A)
U=A-tril(A)
D=diag(diag(A))
x=inv(D)*(b-(L+U)*x0)
```

Escríbela usando el editor de Matlab y guárdala como fichero *jacobi1.m*

Partiendo de (0,0,0) como aproximación inicial de la solución (x0), determina la primera solución del sistema de ecuaciones:

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

Para ello, ejecuta la función, tras asignar por pantalla los valores numéricos a las variables de entrada

1. Escribe una función **jacobi.m**, con estructura *x=jacobi(A,b,x0,tol)*, de modo que resuelva el sistema de ecuaciones **A.x=b** por el método **iterativo de Jacobi** hasta conseguir un error inferior (usar el comando matlab *norm*) a la tolerancia *tol y* partiendo de un valor ***x0*** como aproximación inicial a la solución. La función debe llamar en su interior a la funcion **jacobi1.m** indicada anteriormente.

2. Escribe un script de Matlab que ejecute la función jacobi.m para el sistema

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

con una tolerancia de 0.001. Guarda el script como sistema.m . Este script debe incluir la presentación en pantalla de los pasos o iteraciones sucesivas, mostrando los valores de número de iteración s, los valores de las incógnitas $x_i^{(s+1)}$ (i=1,...n) y el error entre $\mathbf{x}^{(s+1)}$ y $\mathbf{x}^{(s)}$.

SE.5. Utiliza el programa ***jacobi.m***, para resolver el 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.
¿Qué soluciones obtienes?. Si obtienes soluciones incorrectas o no te permite calcular la solución modifica *jacobi.m* para que el programa funcione correctamente

Escribe el conjunto de sentencias que has incorporado al programa ***jacobi.m***, especificando sólo entre qué sentencias se encuentra, es decir escribe la línea anterior, conjunto de sentencias nuevas y líneas posterior que tenías de tu programa de jacobi.m.

SE.6. Teniendo en cuenta tu programa de **jacobi.m** determina la solución del siguiente sistema,

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

¿Qué ocurre?.

SE.7. Escribir una función **x=jacobi_amor(A,b,x0,w,tol)** que resuelva el sistema de ecuaciones **A.x = b** 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.

Aplica tu función al sistema de ecuaciones anterior con una tolerancia de 0.001, un peso de 0.9 y vector solución inicial [0;0;0].

SE.8. Gauss-Seidel (opcional) Escribe, en un anexo, una función llamada **xs1 = gseidel(A,b,xs)** que realice una iteración de Gauss-Seidel sobre el sistema de ecuaciones anterior.