1 Métodos iterativos

En esta práctica se va a estudiar las instrucciones de Matlab para programar los diferentes métodos iterativos.

1.1 Programación de operaciones

Comenzaremos viendo coherencia de operaciones con matrices, y para ello realizaremos funciones simples que sumen o multipliquen matrices. Además utilizaremos esto como repaso de los temas anteriores (condicionales y funciones). También se empleará información sobre el número de argumentos de entrada, a la hora de verificar lo coherencia de la operación.

Consideremos la siguiente función, que suma dos matrices o calcula la inversa

```
function R=opera(A,operacion,B)
%
%Realiza operaciones con una o dos matrices
%Entrada
%   A,B          matrices
%   operacion    suma('+'),resta('-'),multiplicación('*') o inversión(#')
%Salida
%   R            resultado
%
% Verificacion de los argumentos de entrada
switch nargin
    case 2 % La unica operacion admitida es la inversion
        if operacion~='#';error('Dos argumentos y la operacion no es
inversion');end
        [fA,cA]=size(A);
        if fA~=cA;error('La matriz a invertir no es cuadrada');end
        if fA~=rank(A);error('La matriz a invertir no es regular');end
        R=inv(A);
    case 3
        [fA,cA]=size(A);[fB,cB]=size(B);
        switch operacion
            case {'+','-'}
                if fA~=fB|cA~=cB;error('Las matrices no son congruentes');end
                if operacion='-';signo=-1;else;signo=1;end
                R=A+signo*B;
            otherwise
                error('Operación no valida');
        end
    otherwise
        error('Número de argumentos no válido');
end
```

Ejercicios recomendados:

1. Modificar el programa opera.m anterior para que también realice la multiplicación:

El algoritmo para la operación no incluida sería

Paso 1: Caso de operación multiplicación

Paso 2: Si las columnas de la matriz1 no coinciden con las filas de la matriz2,

Paso 3: enviarmensaje de error y finalizar

Paso 4: Calcular el producto de la matriz1 por la matriz2 y almacenarla en R

1.2 Programación de los métodos iterativos

Los métodos de primer orden toman la forma $x^{(m+1)} = \mathbf{K} \cdot x^{(m)} + c$. Dependiendo del método, las matrices de iteración y el vector *c* son diferentes. Así

Método	***K***	*c*
Jacobi	$K = D^{-1}\left(L+U\right)$	$c = D^{-1}b$
G-Seidel	$K = (D-E)^{-1}F$	$c = (D-E)^{-1}b$
SOR	$K = \left(D-\omega L\right)^{-1}\left\{\left(1-\omega\right)D+\omega U\right\}$	$c = \left(D-\omega L\right)^{-1}\omega b$

Tabla I. Matrices de iteración de los diferentes métodos lineales

La obtención de las matrices a partir del sistema es muy simple utilizando las instrucciones ***tril***, ***triu*** y ***diag*** (Se sugiere acudir a la ayuda para una descripción más detallada que la del anexo), y a partir de ellas las correspondientes matrices:

```
» D=diag(diag(A));
» L=-tril(A,-1);
» U=-triu(A,1);
» KJ=D\(L+U) ,cJ=D\b          % Método de Jacobi
» KG=(D-L)\U ,cG=(D-L)\b      % Método de Gauss-Seidel
» w=0.5 ;% Método de relajación con w=0.5
» KS=(D-w*L)\((1-w)*D+w*U), cS=(D-w*L)\(w*b)
```

Sin embargo, para evitar el cálculo de la inversa, su programación se realiza utilizando el siguiente esquema:

Jacobi	$D \cdot x^{(m+1)} = b + L \cdot x^{(m)} + U \cdot x^{(m)}$
G-Seidel	$D \cdot x^{(m+1)} = b + L \cdot x^{(m+1)} + U \cdot x^{(m)}$
SOR	$D \cdot x^{(m+1)} = D \cdot x^{(m)}\left(1-\omega\right) + \omega\left\{b + L \cdot x^{(m+1)} + U \cdot x^{(m)}\right\}$

Tabla II. Algoritmo informático para la resolución de los diferentes métodos iterativos

Se pide:

2. Crear el fichero mitera.m que genere las matrices del método iterativo seleccionado (Jacobi, Gauss-Seidel o relajación) utilizando la instrucción ***switch*** vista en la práctica anterior. Utilizar para su verificación el siguiente sistema:

$$\begin{pmatrix} 3 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 3 \end{pmatrix} X = \begin{pmatrix} -4 \\ 3 \\ 0 \\ 2 \end{pmatrix}$$

$$\text{Jacobi} \rightarrow K = \begin{pmatrix} 0 & -1/3 & 0 & 0 \\ -1/3 & 0 & -1/3 & 0 \\ 0 & -1/3 & 0 & -1/3 \\ 0 & 0 & -1/3 & 0 \end{pmatrix} \wedge c = \begin{pmatrix} -4/3 \\ 1 \\ 0 \\ 2/3 \end{pmatrix}$$

$$\text{Gauss-Seidel} \rightarrow K = \begin{pmatrix} 0 & -1/3 & 0 & 0 \\ 0 & 1/9 & -1/3 & 0 \\ 0 & -1/27 & 1/9 & -1/3 \\ 0 & 1/81 & -1/27 & 1/9 \end{pmatrix} \wedge c = \begin{pmatrix} -4/3 \\ 13/9 \\ -13/27 \\ 67/81 \end{pmatrix}$$

$$\text{Relajación}\left(\omega = 0.5\right) \rightarrow K = \begin{pmatrix} 1/2 & -1/6 & 0 & 0 \\ -1/12 & 19/36 & -1/16 & 0 \\ 1/72 & -19/216 & 19/36 & -1/6 \\ -1/432 & 19/1296 & -19/216 & 19/36 \end{pmatrix} \wedge c = \begin{pmatrix} -2/3 \\ 11/18 \\ -11/108 \\ 227/648 \end{pmatrix}$$

Entrada: variable, matriz A, vector b y variable 'metodo' que selecciona el método a usar. Si fuera relajación habría una variable más que corresponde al valor w. Se debe utilizar la sintaxis (matriz, vector, metodo) ó (matriz, vector, 'relajacion',w) . Los métodos válidos pueden ser: "jacobi"."seidel","gauss","gauss-seidel","relajación"

Salida: K y c, matriz y vector del método iterativo.

La función también debe verificar la coherencia de los datos.

3. Verificar mediante normas y radio espectral, la convergencia de los métodos de Jacobi y Gauss-Seidel con la matriz del ejercicio anterior.
4. Representar, con el mismo sistema, el valor del radio espectral dependiendo del parámetro de relajación. Para ello se hará variar el parámetro entre 0 y 2 con incrementos de 0.1 y se almacena su radio espectral.

2 Anexo: Instrucciones necesarias

2.1.1 Generar tipos especiales de matrices

Función	Operación	Función	Operación
diag(A)	Extraer la diagonal de la matriz A	diag(v)	Matriz diagonal con los elementos de v
tril(A)	Parte triangular inferior de la matriz A	triu(A)	Parte triangular superior de la matriz A
eye(n)	Crea la matriz identidad de orden n-	eye(m,n)	Idem orden mxn.
zeros(n)	Crea la matriz nula de orden n	zeros(m,n)	Idem de orden mxn
ones(n)	Crea la matriz de unos de orden n	ones(m,n)	Idem de orden mxn
rand(n)	Crea una matriz aleatoria uniforme de orden n	rand(m,n)	Idem de orden mxn

2.2 Operaciones Lógicas con Matrices

any(V)	Devuelve 0 si todos los elementos del vector V son nulos, y devuelve 1 si algún elemento de V es no nulo
all(V)	Devuelve 1 si todos los elementos del vector V son no nulos , y devuelve 0 si algún elemento de V es nulo
find(V)	Devuelve los lugares (o índices) que ocupan los elemento no nulos del vector V

2.3 Resolución de Sistemas y factorización

Función	Operación	Función	Operación
X=A\B	Resuelve el sistema A*X=B	X=A/B	Resuelve el sistema X*A=B
lu(A)	Factorización LU	chol(A)	Factorización de Choleski

2.3.1 Autovalores y Autovectores

Función	Operación	Función	Operación
eig(A)	Autovalores de la matriz A	[V,D]=eig(A)	M matriz diagonal de autovalores D y matriz V de autovectores por columnas
poly(A)	Polinomio característico de A	poly(V)	Vector cuyas raíces son los elementos del vector V

Otras operaciones de matrices y vectores

transpose(A)	Traspuesta de la matriz A	det(A)	Determinante de la matriz A
A'	Traspuesta de la matriz A	trace(A)	Suma de los elementos de la diagonal de A
inv(A)	Inversa de la matriz A (A-1)	norm(A)	Norma de A (mayor valor singular de la matriz A)
size(A)	Filas y columnas de la matriz A	length(x)	Longitud del vector x
max(x)	Máximo de los elementos de x	min(x)	Mínimo de los números los elementos de x
abs(z)	Módulo o valor absoluto del valor z		