UNIVERSIDAD DE CONCEPCION
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA MATEMATICA

Cálculo Numérico (521230) - Laboratorio 4

Sistemas de Ecuaciones Lineales: Métodos Iterativos

Ejercicio 1. (*Trabajo con matrices sparse*) Escriba una función MATLAB que, dados valores $n \in \mathbb{N}, a, b, c, d \in \mathbb{R}$, retorne la matriz $A \in \mathbb{R}^{n\times n}$ con la siguiente estructura

$$A = \begin{pmatrix} a & b & 0 & 0 & \cdots & 0 & d & 0 & 0 \\ c & a & b & 0 & \cdots & 0 & 0 & d & 0 \\ 0 & c & a & b & \cdots & 0 & 0 & 0 & d \\ 0 & 0 & c & a & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ d & 0 & 0 & 0 & \cdots & c & a & b & 0 \\ 0 & d & 0 & 0 & \cdots & 0 & c & a & b \\ 0 & 0 & d & 0 & \cdots & 0 & 0 & c & a \end{pmatrix}.$$

Nombre a su función `matriz_preg1_lab4.m`. Note que muchos de los elementos de la matriz anterior son iguales a cero, ella es una matriz dispersa.

1.1 Llame a la función por usted escrita con $n = 100, a = 4, b = c = 2, d = 7$. Guarde el resultado en la variable `A`.

1.2 Los comandos en MATLAB `spy` y `nnz` sirven verificar si una matriz es dispersa. Úselos con la matriz creada en el inciso anterior.

1.3 La matriz A puede almacenarse en MATLAB en un formato especial denominado *sparse*. Esto puede lograrse de 2 formas:

- transformando, mediante `B = sparse(A)`, la matriz creada en **1.1** a formato *sparse*,
- usando comandos que permitan crear directamente una matriz *sparse*. Vea en `matriz_preg1_lab4_sparse.m` una forma de crear A directamente como matriz *sparse*. Abra `matriz_preg1_lab4_sparse.m` y, con ayuda del comando `help` de MATLAB, trate de entender la forma en que se usan en ella los comandos `sparse` y `spdiags`.

Escriba en la ventana de comandos MATLAB

```
>> B = sparse(A);
>> C = matriz_preg1_lab4_sparse(100,4,2,2,7);
>> whos
```

1.4 Abra el rutero `preg1_lab4.m`.

El objetivo de este rutero es comparar los tiempos de ejecución al multiplicar una matriz dispersa por sí misma, almacenada en forma *full* (guardando todas las entradas de la matriz, también aquellas que son iguales a cero) y en forma *sparse*. Llame al rutero y observe el gráfico que se genera. ¿Con cuál de las dos formas de almacenamiento requiere el producto de una matriz dispersa por sí misma menos tiempo computacional?

Ejercicio 2. (*Jacobi, Gauss Seidel, Gradiente Conjugado*) Abra la función `jacobisol.m`. Almacene esta función en `gaussseidelsol.m` y modifíquela de forma que permita obtener una aproximación a la solución exacta de un sistema de ecuaciones lineales mediante el método de Gauss-Seidel.

Escriba un rutero MATLAB en el que haga lo siguiente:

- Llame a la función `matriz_preg1_lab4_sparse` con $n = 50, a = 4, b = c = 1, d = -1$.
- Calcule, para la matriz creada antes, las matrices D, E y F de la descomposición introducida en clases para derivar los métodos de Jacobi y Gauss-Seidel. Compruebe que las matrices D, E y F también se almacenan como matrices *sparse*.
- Calcule las correspondientes matrices de iteración de los métodos de Jacobi (J) y Gauss-Seidel (GS). Dado que la matriz creada antes es estrictamente diagonal dominante, el radio espectral de las matrices J y GS debe ser estrictamente menor que uno. Compruébelo con ayuda de los comandos `eigs`, `abs` y `max`.
- Llame a las funciones `jacobisol` y `gaussseidelsol` para resolver el sistema $Ax = b$ con A construida al inicio de este ejercicio y b igual al vector cuyas 50 componentes son iguales a uno. Tome $x^{[0]} = \Theta$, `maxiter=100` y `tol` según se especifica en el siguiente cuadro que usted debe completar y en el que x_J denota la aproximación obtenida mediante el método de Jacobi y x_{GS}, la obtenida con Gauss-Seidel.

	iteraciones Jacobi	iteraciones Gauss-Seidel	$\|x_J - x_{GS}\|_\infty$
`tol=1e-2`			
`tol=1e-4`			
`tol=1e-6`			

Note que en este caso particular el método de Gauss-Seidel necesita menos iteraciones que el método de Jacobi para alcanzar la precisión requerida. Recuerde que éste no siempre es el caso.

Escriba un segundo rutero MATLAB en el que haga lo siguiente:

- Llame a la función `matriz_preg1_lab4_sparse` con $n = 50, a = 4, b = c = 1, d = -1$.
- Compruebe que esta matriz no es mal condicionada.
- Utilice el comando `pcg` para resolver el sistema $Ax = b$ (siendo A la matriz creada antes y b el vector con 50 componentes iguales a 1) con ayuda del método del gradiente conjugado (haciendo `help pcg` podrá ver cómo debe llamarse esta función). Complete el siguiente cuadro:

	iteraciones GC	$\frac{\|b - Ax_{gc}\|}{\|b\|}$
`tol=1e-2`		
`tol=1e-4`		
`tol=1e-6`		

- Genere mediante `A = gallery('poisson',100)` una matriz de orden 10000 cuyo número de condición es mucho mayor que el de la matriz en los incisos anteriores (compruébelo haciendo `condest(A)`). Resuelva el sistema $Ax = b$ con `b = ones(10000,1)` y el método del gradiente conjugado con `tol = 1e-3`. ¿Logra el método calcular una aproximación aceptable a la solución exacta de $Ax = b$?

Llame al método del gradiente conjugado precondicionado con distintos precondicionadores y `tol=1e-3`. El precondicionador puede generarse mediante un llamado a `ichol` o a `cholinc` (en última versión de MATLAB se recomienda el uso de `ichol`) como sigue:

```
>> tolp = 0.2;
>> L = ichol(A,struct('type','ict','droptol',tolp));
>> P = L*L';
```

o como sigue:

```
>> tolp = 0.2;
>> R = cholinc(A,tolp);
>> P = R'*R;
```

Complete el siguiente cuadro, en él P es el precondicionador calculado en cada caso.

	iteraciones GCP	$\|P - A\|_\infty$
`tolp=0.2`		
`tolp=0.01`		
`tolp=0.001`		

Ejercicio 3. (*Google Matrix*) Actualmente google se ha establecido como la página de búsqueda en internet más utlizada. Esto ha sido en gran parte gracias al novedoso algoritmo que esta compañía utiliza para, una vez encontradas las páginas que cumplen con el criterio de búsqueda de un usuario, escoger el orden en que éstas serán presentadas. Para escoger este orden google asocia a cada página un número, llamado `pagerank`, que calcula teniendo en cuenta el número de enlaces desde y hacia cada una de las páginas web en el mundo. El orden en que se presentan las páginas depende de su `pagerank` (las de mayor `pagerank` se presentan primero). Para calcular el `pagerank` de cada página web, google crea una matriz G que tiene una fila y una columna por cada página web, las entradas de esta matriz son

$$G(i,j) = \begin{cases} 1, & \text{si hay un enlace de página } j \text{ a página } i, \\ 0, & \text{en caso contrario.} \end{cases}.$$

El `pagerank` de una página depende no sólo del número de páginas que contengan enlaces a ella, sino también del `pagerank` de esas páginas. Así una página que sea referenciada en muchas páginas de poca importancia podrá tener menos importancia que una que sea referenciada en pocas páginas de mucha importancia. Si x es un vector con el `pagerank` de cada una de las páginas web del mundo, google lo encuentra resolviendo el siguiente sistema de ecuaciones lineales

$$Ax = e, \quad A = I - pGD \tag{1}$$

dónde I es la matriz de identidad, p es la probabilidad de que una persona, en su búsqueda de alguna información en internet, siga uno de los enlaces en la página donde se encuentra y se toma igual a 0,85, G es la matriz mencionada antes, D es una matriz diagonal tal que $D(i,i)$ es el número de enlaces en la página i y e es el vector que contiene todas sus entradas iguales a 1. Encontremos experimentalmente, de entre todos los algoritmos que hemos estudiado para la solución de sistemas de ecuaciones lineales, cuál es el más conveniente para resolver este sistema de ecuaciones.

3.1 Otros archivos en `lab4_2015_1.zip` son `elsurdata.mat`, `creatematrix_pagerank.m` y `mostrarrangos.m`. Cargue las variables almacenadas en `elsurdata.mat` en MATLAB. Para ello usted debe escribir en la ventana de comandos de MATLAB

```
>> load elsurdata
```

Escriba

```
>> whos
```

y observe que la matriz `Gelsur` y el vector `Uelsur` se han cargado en memoria (éstas eran las variables guardadas en `elsurdata.mat`). La matriz `Gelsur` es como la matriz G mencionada antes, pero se creó suponiendo que la web está formada sólo por 500 páginas (`www.elsur.cl`, los enlaces contenidos en ella , los enlaces en esos enlaces y así sucesivamente, 3 direcciones fueron borradas de la lista por ser no válidas). La variable `Uelsur` contiene los nombres de las 497 páginas consideradas en la red que parte en `www.elsur.cl`.

Escribiendo `A = creatematrix_pagerank(Gelsur)` se construye la matriz A en (1).

3.2 Con `spy(A)` observe la estructura de A. ¿Es una matriz dispersa? ¿Es simétrica? Determine, mediante un llamado a `lu`, las matrices L y U de la descomposición LU de A. ¿Cuántos elementos distintos de cero tiene A? ¿Cuántos tienen L y U?

3.3 Busque una aproximación a la solución exacta a este sistema de ecuaciones con los métodos iterativos que usted conoce. ¿Cuáles métodos iterativos le permitieron obtener una aproximación a la solución exacta de (1)? ¿Cuál tolerancia usó? ¿En cuántas iteraciones alcanzó cada uno de ellos la precisión requerida? ¿Cuál es la diferencia entre las soluciones obtenidas? Si alguno de los métodos no le permite obtener una aproximación con la precisión requerida, ¿a qué se debe esto?

3.4 Una vez encontrada una aproximación a la solución exacta d e (1) usted podrá, llamando a la función `mostrarrangos`, ver un código de barras con los rangos de las 10 páginas que tienen mayor `pagerank`, así como una lista con sus direcciones, número de enlaces en y hacia ellas y el `pagerank` de cada una.