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

---

# Cálculo Numérico (521230)

## Laboratorio 4
## Sistemas de ecuaciones lineales – II

El objetivo de este laboratorio es aprender a utilizar eficientemente métodos **iterativos** para la solución de sistemas de ecuaciones lineales $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, así como el correcto tratamiento de matrices dispersas (*sparse*).

1. (a) Haga un programa *function* que genere una matriz de la forma

$$\boldsymbol{B} = \begin{pmatrix} 2\,\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & 2\,\boldsymbol{I} \end{pmatrix} \in \mathbb{R}^{2n\times 2n}$$

para $n = 1000$. Compruebe que $\boldsymbol{B}$ es una matriz dispersa mediante el comando `nnz` (*Number of Non Zeros*) que da la cantidad total de entradas no nulas de la matriz.

(b) Utilice la siguiente sentencia para almacenar en forma *sparse* otra matriz $\boldsymbol{A}$ igual a $\boldsymbol{B}$:

```
A=sparse(B);
```

Utilice el comando `whos` para determinar la cantidad de memoria que requiere el almacenamiento de cada una de esas matrices. Indique cuál resulta más conveniente.

(c) Genere un vector aleatorio $\boldsymbol{b} \in \mathbb{R}^{2n}$ y compare los tiempos necesarios para resolver los sistemas $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{b}$ y $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Indique cuál resulta más conveniente.

(d) Hay varios comandos que tienen su contraparte para matrices dispersas. Por ejemplo, `speye` es semejante a `eye`, pero la matriz identidad que genera se almacena como *sparse*.

Al realizar operaciones con matrices *sparse* (por ejemplo, suma, producto, traspuesta, \, concatenación, etc.), MATLAB almacena los resultados en nuevas matrices *sparse*.

Así mismo, muchos comandos, cuando se aplican a matrices *sparse*, generan matrices *sparse*. Por ejemplo, `diag`, `tril` y `triu`.

Utilice estos comandos para obtener las matrices $\boldsymbol{D}$, $\boldsymbol{L}$ y $\boldsymbol{U}$ de la descomposición $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}$. Calcule la matriz de iteración del método de Jacobi $\boldsymbol{J} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$. Compruebe que las matrices $\boldsymbol{D}$, $\boldsymbol{L}$, $\boldsymbol{U}$ y $\boldsymbol{J}$ tambien se almacenan como matrices *sparse*.

(e) El comando `spy` permite "espiar" la estructura de una matriz mediante un gráfico con la distribución de sus entradas no nulas. "Espíe" la estructura de las matrices $\boldsymbol{A}$, $\boldsymbol{D}$, $\boldsymbol{L}$, $\boldsymbol{U}$ y $\boldsymbol{J}$.

(f) Si bien el almacenamiento de la matriz $\boldsymbol{A}$ requiere mucha menos memoria que el de la matriz $\boldsymbol{B}$, para generar $\boldsymbol{A}$ como se ha descrito resulta necesario haber almacenado previamente $\boldsymbol{B}$.

A veces, la memoria del computador no alcanza para almacenar $\boldsymbol{B}$ en forma llena. En tal caso, es necesario generar directamente la matriz dispersa $\boldsymbol{A}$ sin pasar nunca por la matriz llena $\boldsymbol{B}$. Indique alguna forma de hacer esto y compruébelo.

(g) Para algunas aplicaciones es necesario dar la forma llena de una matriz dispersa. El comando inverso de `sparse` que almacena una matriz dispersa en forma de matriz llena es `full`. Compruébelo.

2. (a) Haga un programa que resuelva mediante el método de **Jacobi** un sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ con error menor que una tolerancia dada `tol`. Parta de un dato inicial $\boldsymbol{x}^0$ nulo y utilice el criterio de detención estudiado en las clases teóricas. El programa también debe detenerse si se alcanza un número máximo de iteraciones `maxit` sin que se satisfaga el criterio de convergencia.

   Testee el método con la matriz del Ej. 1(b) del Lab. 3 con una tolerancia `tol` $= 10^{-6}$. Verifique que la solución obtenida aproxima a la solución exacta con la tolerancia del error estipulada. Determine la cantidad de iteraciones realizadas.

   (b) Repita lo anterior con el método de **Gauss-Seidel**. Indique cuál de los dos métodos requiere menos iteraciones.

3. Baje de la página web del curso los archivos de datos `data-1.mat`, `data-2.mat` y `data-3.mat` y el programa `show.m`. En los archivos se han almacenado mediante un comando `save` una matriz $\boldsymbol{A}$ (en forma *sparse*) y un vector $\boldsymbol{b}$ generados mediante un programa MATLAB para la resolución por el método de **elementos finitos** de un problema de EDPs que modela la deformación de una membrana bajo la acción de una fuerza. Estos archivos contienen también datos geométricos para la visualización de los resultados mediante el programa `show.m`.

   Cada uno de los tres archivos de datos corresponde a modelaciones de distintos grados de precisión de la membrana: `data-1.mat` es una modelación grosera, `data-2.mat` una normal y `data-3.mat` una bien fina. Por ello las dimensiones de las matrices y vectores son sustancialmente distintas.

   (a) Cargue el archivo `data-1.mat` mediante un comando `load`. Determine las dimensiones del problema y "espíe" la matriz $\boldsymbol{A}$.

   (b) Resuelva el sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ correspondiente al archivo `data-1.mat` mediante el programa de Gauss-Seidel del ejercicio anterior, con error menor que $10^{-6}$. Indique cuántas iteraciones son necesarias.

   (c) Aplique el método del gradiente conjugado mediante el comando `pcg`. La sentencia para la aplicación elemental de este comando es la siguiente:

   ```
   >> x=pcg(A,b);
   ```

   Indique lo qué ocurre.

   (d) Utilice el comando `help` para ver como pueden modificarse la tolerancia del error y el número máximo de iteraciones en el comando `pcg` . Aplique otra vez el método del gradiente conjugado con tolerancia $10^{-6}$ y con un número de iteraciones suficiente para lograr la convergencia.

   (e) Visualice la deformación calculada de la membrana ejecutando el programa `show`. (Ojo! El programa presupone que la solución calculada está en un vector llamado `x`.)

   (f) Repita los pasos anteriores (a), (d) y (e) con los otros dos archivos.