Universidad de Concepción
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

Laboratorio 2: Introducción a Octave. Parte II.
Cálculo Numérico (521230)

Observaciones

- En esta guía se plantean tres problemas.
  - El primero se resuelve en el video asociado al laboratorio. Los archivos `.m` escritos pueden ser descargados de Canvas.
  - **El segundo problema debes entregarlo**.
  - El problema tres es **solo** para tu trabajo personal, **no** debe ser entregado, sirve para complementar tu formación en el curso.
- Te recomendamos leer esta guía antes de ver el video de resolución del problema.

La semana pasada vimos como definir vectores y matrices en Octave y como graficar funciones ya incorporadas a Octave. La manera en que trabajamos la semana pasada con Octave (escribiendo directamente nuestros comandos en la ventana de comandos) puede ser incómoda si uno quiere resolver problemas más complejos.
Esta semana continuaremos trabajando con matrices y vectores y veremos además como escribir ciclos en Octave y como definir nuevas funciones y ruteros (scripts).

Supongamos que deseamos calcular el producto interior entre los vectores $\left(\frac{1}{2}, -\frac{3}{5}, \frac{1}{7}\right)^{\mathrm{T}}$ y $\left(3, \sqrt{2}, \frac{1}{3}\right)^{\mathrm{T}}$. Tenemos varias formas de hacerlo en Octave, una de ellas es escribir

Producto interior

```
>> p = (1/2)*(3) + (-3/5)*sqrt(2) + (1/7)*(1/3)
```

pero, si queremos además calcular el producto interior de los vectores $\left(\frac{1}{3}, -1, \frac{2}{3}, -3\right)^{\mathrm{T}}$ y $\left(3, \sqrt{2}, \frac{1}{3}, 10\right)^{\mathrm{T}}$, debemos escribir toda la suma de nuevo. Es mucho más sencillo escribir una función que, dados dos vectores cualesquiera $x, y \in \mathbb{R}^n$, retorne el producto interior de x e y.

Octave **en PC:** Si has instalado Octave en tu computador, puedes abrirlo y hacer click en
`File -> New Function`.
Octave te pregunta el nombre que quieres darle a la función, en mi caso la llamé `productointerior` y te sugiero darle el mismo nombre pues haremos referencia a él varias veces. Se abre entonces, en el editor de Octave, un archivo cuyas primeras líneas son comentarios (comienzan con `%`, no nos interesan por ahora). Al final de esas líneas aparece lo siguiente

```
function retval = productointerior (input1, input2)


endfunction
```

En la carpeta de trabajo con Octave hay ahora un nuevo archivo cuyo nombre es `productointerior.m`.
En la función recién creada `input1`, `input2` representan parámetros de entrada y `retval`, un parámetro de salida. Podemos cambiar el nombre a los parámetros de entrada y dar otro nombre a la salida. Llamemos `x` e `y` a los vectores de entrada y `prod` a la salida.

```
function prod = productointerior (x, y)


endfunction
```

Octave **en línea:** Si decides trabajar con Octave en línea, puedes crear un archivo nuevo, haciendo click en + y darle el nombre `productointerior.m`. Se crea un archivo en cuya primera línea aparece

`disp(``Hello World'')`, bórrala o coméntala (escribiendo `%` delante de ella) y escribe

```
function prod = productointerior (x, y)


endfunction
```

Comencemos a escribir la función `productointerior` (tanto si hemos instalado OCTAVE en el PC como si estamos trabajando con la versión en línea del software).

1) Los vectores `x` e `y` deben tener la misma cantidad de elementos, de lo contrario, no podremos calcular su producto interior.
   Para comprobar si tienen la misma cantidad de elementos llamamos a la función `length`, que retorna la cantidad de elementos de un vector, y utilizamos la construcción `If` de OCTAVE para retornar un mensaje de error si `length(x) ~= length(y)` (longitud de `x` es distinto de longitud de `y`).
   La función `error` es una función OCTAVE que nos permite parar la ejecución de un programa devolviendo un mensaje de error. Más adelante veremos como funciona.

```
function prod = productointerior (x,y)
if length(x) ~= length(y)
  error('vectores de entrada deben tener la misma cantidad de componentes')
else
endif
endfunction
```

   Ahora debemos completar qué va a hacer nuestra función cuando los vectores de entrada tengan la misma cantidad de elementos, es decir, las instrucciones entre las palabras `else` y `endif`.
2) Si $x, y \in \mathbb{R}^n$ y x_i denota la i-ésima componente de x, el producto interior de x e y es el número real

$$x_1y_1 + x_2y_2 + \ldots + x_ny_n.$$

   La forma más sencilla de calcular esta suma en OCTAVE es con un un ciclo `for`, haciendo

```
prod = 0;
n = length(x);
for i = 1 : n
   prod = prod + x(i)*y(i);
endfor
```

   Con el ciclo `for` anterior la variable `i` va a tomar los valores $1, 2, \ldots, n$. Recuerda que con `1:n` se crea un vector fila cuyas componentes son $1, 2, \ldots, n$, con la construcción `for i = 1:n` estamos indicando que i va a tomar el valor de cada una de las componentes de este vector y que las instrucciones entre `for` y `endfor` se realizarán para cada valor de i.
   Podemos entonces completar la función que estamos escribiendo:

```
function prod = productointerior (x,y)
if length(x) ~= length(y)
  error('vectores de entrada deben tener la misma cantidad de componentes')
else
 prod = 0;
 n = length(x);
 for i = 1:n
   prod = prod + x(i)*y(i);
 endfor
endif
endfunction
```

Nota que hemos guardado el producto interior en una variable llamada `prod` porque `prod` es el nombre que dimos al parámetro de salida de la función.

3) Ahora podemos llamar a la función. Regresemos a la ventana de comandos de OCTAVE y escribamos

Llamando a productointerior

```
>> u = [1,sqrt(2),-3/2,1/4]; v = [1 2 3 4];
>> productointerior(u,v)
```

El resultado se asigna a `ans`.

Si escribimos

Llamando a productointerior

```
>> u = [1,1,1,1]; v = [1 2 3 4];
>> p = productointerior(u,v)
```

el resultado se asigna a `p`.

¿Qué ocurre cuando escribimos lo siguiente en la ventana de comandos?

Llamando a productointerior

```
>> u = [1,1,1]; v = [1 2 3 4];
>> p = productointerior(u,v)
```

En este caso, como los vectores `u` y `v` no tienen la misma cantidad de elementos, en lugar de realizar el ciclo `for`, nuestro programa `productointerior` llama a la función `error` con lo que se detiene la ejecución del mismo mostrando nuestro ensaje de error.

Llamemos a la función recién escrita para calcular el producto interior de los siguientes vectores

$$2^{512}\begin{pmatrix}1\\2\end{pmatrix} \quad \text{y} \quad 2^{512}\begin{pmatrix}3\\4\end{pmatrix}.$$

Para ello escribimos en la ventana de comandos de OCTAVE

Llamando a productointerior

```
>> u = 2^(512)*[1,2]; v = 2^(512)*[3 4];
>> p = productointerior(u,v)
```

¿Qué respuesta obtenemos? La respuesta es `Inf` porque el resultado de multiplicar 2^{512} y $3 \cdot 2^{512}$ es un número mayor que el mayor número que puede ser almacenado por OCTAVE en el computador.

Un par de comentarios antes de continuar:

1) Todos los programas OCTAVE deben guardarse en archivos con extensión `.m`.
2) Un programa debe ser ejecutado desde la ventana de comandos y el archivo `.m` debe estar ubicado en el directorio de trabajo actual.
3) En un programa OCTAVE todo lo que siga al símbolo `%` es interpretado como un comentario.

4) Existen dos tipos de programas en OCTAVE: ruteros (scripts) y funciones. Una función permite parámetros de entrada y salida, un rutero no.
5) Un rutero (o script) es una "recopilación" ordenada de comandos que pueden ser ejecutados desde la ventana de comandos al escribir en ella el nombre del rutero.
6) La primera línea del programa en una función tiene que ser de la forma
   `function [salida] = <nombre>(entrada)`
   `salida` es una lista con los nombres de los parámetros de salida de la función separados por comas, mientras que `entrada` es una lista con los nombres de todos los parámetros de entrada a la función, también separados por comas.
7) Cada nueva función que escribamos debe estar en un nuevo archivo `.m` cuyo nombre debe ser igual al de la función en él.

Supongamos ahora que deseamos llamar a la función `productointerior` recién escrita para calcular los productos interiores de los siguientes pares de vectores x, y:

$$x, y \in \mathbb{R}^{10} : x = \Big(1, 1, \ldots, 1\Big), \qquad y = \Big(1, 2, \ldots, 10\Big),$$
$$x, y \in \mathbb{R}^{20} : x = \Big(1, -1, 1, \ldots, 1, -1\Big), \qquad y = \Big(2, 2, \ldots, 2\Big),$$
$$x, y \in \mathbb{R}^{100} : x = \Big(0, 0, \ldots, 0, 1, 0, \ldots, 0\Big), \qquad y = \Big(1, 2, \ldots, 100\Big)$$

donde en el último vector x el número 1 está en la posición 50.
Para ello escribamos un rutero (script) que llame tres veces a la función que escribimos antes.

OCTAVE **en PC:** Si has instalado OCTAVE en tu computador, puedes hacer click en
`File -> New Script`.
Se abre entonces, en el editor de OCTAVE, un archivo en blanco.

OCTAVE **en línea:** Si decides trabajar con OCTAVE en línea, puedes crear un archivo nuevo, haciendo click en + y darle el nombre `llamados_pi.m`. Se crea un archivo en cuya primera línea aparece `disp(``Hello World'')`, bórrala o coméntala (escribiendo `%` delante de ella).

Escribamos los siguientes comandos en el rutero

Llamando a productointerior
```
x = ones(1,10); y = 1:10;
p1 = productointerior(x,y);
x = ones(1,20); x([2:2:20]) = -1; y = 2*ones(1,20);
p2 = productointerior(x,y);
x = zeros(1,100);x(50)=1; y = 1:100;
p3 = productointerior(x,y);
```

y guardémoslo con el nombre `llamados_pi.m`.
Ahora escribamos en la ventana de comandos de OCTAVE o de OCTAVE en línea,

Llamando a productointerior
```
>> llamados_pi
```

Los valores de `p1`, `p2` y `p3` están en la memoria de trabajo de OCTAVE. Puedes mirarlos escribiendo `p1` o `p2` o `p3` en la ventana de comandos de OCTAVE.
Un par de observaciones en relación a los comandos que escribimos en el rutero `llamados_pi.m`.

1) Los comandos `ones` y `zeros` permiten crear matrices con todas sus componentes iguales a 1 (`ones`) o 0 (`zeros`).
2) Con `2:2:20` OCTAVE crea un vector cuya primera componente es 2, la última es 20 y la diferencia entre dos componentes consecutivas es 2, es decir, crea el vector `[2 4 6 8 ... 20]`. Al escribir `x([2:2:20]) = -1` estamos asignando el valor -1 a las componentes $2, 4, 6, \ldots, 20$ de `x`.

Problema resuelto en video:

Sea $a \in \mathbb{R}^+$. La sucesión que, dado $x_0 \in \left]0, \frac{2}{a}\right[$, se define mediante

$$x_n = x_{n-1}\left(2 - a x_{n-1}\right), \text{ para } n \in \mathbb{N}$$

converge a $\dfrac{1}{a}$.

1) Escriba una función, `sucesionlab2.m` que, dados a, x_0 y $n \in \mathbb{N}$ haga lo siguiente:
   - Compruebe si $x_0 \in \left]0, \dfrac{2}{a}\right[$ y $a > 0$. Si alguna de las condiciones anteriores no se cumple, debe retornar con un mensaje de error.
   - Si $x_0 \in \left]0, \dfrac{2}{a}\right[$ y $a > 0$, calcule n términos de la sucesión definida antes y retorne un vector que los contenga y cuya primera componente sea x_0.
2) Escriba un rutero `calculasucesion.m` que:
   - Llame a la función anterior con $a = 2$, $x_0 = 0.9$ y $n = 5$.
   - Grafique los elementos de la sucesión calculada, es decir grafique los pares $(i, x_i),\ i = 0, 1, 2, \ldots, n$.
   - Grafique, en el mismo gráfico anterior, la recta $y = \dfrac{1}{a}$, para comprobar si los elementos de la sucesión se acercan a $\dfrac{1}{a}$.
   - Determine el error absoluto que se comete al aproximar $\dfrac{1}{a}$ por el último término calculado de la sucesión.
   - Repita los ítemes anteriores con $a = 0.01$, $x_0 = 2$ y $n = 5$.

Problema a entregar: Sea $a \in \mathbb{R}^+$. La sucesión que, dado $x_0 \in \mathbb{R}^+$, se define mediante

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right), \text{ para } n \in \mathbb{N}$$

converge a $\sqrt{a}$.

1) Escriba la función, `sucesionlab2.m` que, dados a, x_0 y $n \in \mathbb{N}$ haga lo siguiente:
   - Compruebe si $x_0 > 0$ y $a > 0$. Si alguna de las condiciones anteriores no se cumple, debe retornar con un mensaje de error.
   - Si ambas condiciones se cumplen, calcule n términos de la sucesión definida antes y retorne un vector que los contenga y cuya primera componente sea x_0.
2) Escriba el rutero `calculasucesion.m` que:
   - Llame a la función anterior con $a = 2$, $x_0 = 5$ y $n = 10$.
   - Grafique los elementos de la sucesión calculada, es decir grafique los pares $(i, x_i),\ i = 0, 1, 2, \ldots, n$.
   - Grafique, en el mismo gráfico anterior, la recta $y = \sqrt{a}$, para comprobar si los elementos de la sucesión se acercan a $\sqrt{a}$.
   - Determine el error absoluto que se comete al aproximar $\sqrt{a}$ por el último término calculado de la sucesión.
   - Repita los ítemes anteriores con $a = 0.01$, $x_0 = 2$ y $n = 10$.

Forma de entrega:

- Dos archivos .m (`sucesionlab2.m` y `calculasucecion.m`).

Problema adicional para el trabajo con OCTAVE , no debes entregarlo:

El **proceso de ortogonalización de Gram-Schmidt** es un procedimiento, aprendido en Álgebra 2, que permite, dado un conjunto de vectores linealmente independiente $\{v_1, v_2, \ldots, v_m\}$ de cierto espacio vectorial V, determinar un conjunto ortogonal $\{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_m\}$ que además satisface

$$\langle \{v_1, v_2, \ldots, v_m\} \rangle = \langle \{\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_m\} \rangle .$$

Los vectores $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_m$ se calculan de la siguiente forma:

$$\begin{aligned}
\hat{v}_1 &= v_1, \\
\hat{v}_2 &= v_2 - \frac{\langle v_2, \hat{v}_1 \rangle}{\|\hat{v}_1\|_2^2} \hat{v}_1, \\
\hat{v}_3 &= v_3 - \frac{\langle v_3, \hat{v}_1 \rangle}{\|\hat{v}_1\|_2^2} \hat{v}_1 - \frac{\langle v_3, \hat{v}_2 \rangle}{\|\hat{v}_2\|_2^2} \hat{v}_2, \\
&\vdots \\
\hat{v}_m &= v_m - \sum_{j=1}^{m-1} \frac{\langle v_m, \hat{v}_j \rangle}{\|\hat{v}_j\|_2^2} \hat{v}_j.
\end{aligned}$$

1) Modifique el procedimiento anterior para que el vector $\hat{v}_i$ que se calcula en cada paso sea un vector unitario (con norma 2 igual a 1).
2) Escriba la función OCTAVE `GS.m` que, dada cierta matriz $A \in \mathcal{M}_{m\times n}(\mathbb{R})$, retorne la matriz $B \in \mathcal{M}_{m\times n}(\mathbb{R})$ cuyas columnas son las que resultan de aplicar el procedimiento anterior a las columnas de A.
3) Justifique por qué la matriz $B \in \mathcal{M}_{m\times n}(\mathbb{R})$ resultante debería satisfacer
$$B^{\mathrm{T}} B \approx I.$$
4) Modifique `GS.m` para que, además de retornar B, retorne el valor de
$$\max_{1\leq i\leq n} \|(B^{\mathrm{T}}B)(:,i) - I(:,i)\|_2,$$
donde $I(:,i)$ es la i-ésima columna de I y $(B^{\mathrm{T}}B)(:,i)$, la i-ésima columna de $B^{\mathrm{T}}B$. Este segundo parámetro de salida es una medida de la distancia entre $B^{\mathrm{T}}B$ e I.
5) Llame a `GS.m` con las siguientes matrices
$$A = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}$$
y $\varepsilon \in \{10^{-4}, 10^{-6}, 10^{-8}\}$.
6) Observe que a menor valor de ε, más se aleja $B^{\mathrm{T}}B$ de la matriz identidad. El proceso de ortogonalización de Gram-Schmidt, a pesar de ser un procedimiento teóricamente correcto, no siempre da, al implementarse en un computdor, los resultados esperados.