UNIVERSIDAD DE MURCIA

Departamento de Matemáticas
Cálculo Numérico en una Variable
Curso 2021/22

Prof. S. Sánchez-Pedreño

Práctica 2
3 de octubre de 2021

El objetivo principal de esta práctica es el de familiarizarnos con las herramientas gráficas de Octave: dibujo de gráficas en el plano, diferentes tipos de línea en el gráfico, colores, leyendas, etc.

Utilizaremos, con ese objetivo, algunas funciones sencillas, pero también intentaremos mediante el último ejercicio, obtener más experiencias acerca de los problemas de inestabilidad o de pérdida de cifras significativas en algunos cálculos numéricos.

Introducción: herramientas gráficas de Octave

Recordemos que Octave (o Matlab) trabajan fundamentalmente con vectores y matrices, es importante tenerlo en cuenta, puesto que para dibujar una curva en el plano lo que haremos (con la orden gráfica más elemental) será pasarle un vector de abscisas y un vector de ordenadas. Con esos datos Octave dibujará una poligonal (que puede ser «invisible») que une los puntos del plano dados por las componentes de ambos vectores: eso será una curva gráfica para Octave.

Comenzamos por tanto con algunas herramientas para crear o manejar vectores o matrices. Recordemos que los índices de las componentes de un vector o matriz se numeran empezando en 1, no en 0 (al contrario que en Java u otros lenguajes).

Orden	**Descripción**	**Ejemplo (a teclear)**
`[*,*,*;*,*,*]`	Crea una matriz descrita por valores concretos (`*`). Las columnas se separan mediante una coma, las filas mediante punto y coma (`;`)	`>> A=[1,2,3;9,10,11]` `>> B=[2,4,6,8]`
`A(n,m)`	Devuelve el elemento de la matriz `A` en la fila `m` y la columna `n`. Naturalmente para vectores sólo necesitamos una posición. Recuerda: `B(0)` producirá un error.	`>> A(2,3)` `>> B(2)`
`ini:incr:max` `[ini:incr:max]`	Crea un vector cuyo primer elemento es `ini` y los sucesivos se obtienen sumando `incr` al anterior, hasta alcanzar el valor `max`. Si el valor max se sobrepasa (porque al incrementar sucesivamente no lo obtenemos exactamente) el último elemento será menor que `max`.	`>> A=[2:3:32]` `>> B=[1:0.1:2.005]`
`linspace(ini,fin,n)`	Crea un vector con `n` componentes cuyos valores están equidistribuidos en el intervalo `[ini,fin]`.	`>> linspace(1,10,10)` `>> linspace(pi,pi+1,5)`

`zeros(n,m)`
`zeros(n)`
`zeros(...,"clase")`
	Crea una matriz de ceros. O bien de tamaño `n`×`m`, o `n`×`n`. Las componentes serán de tipo `clase`; si no se incluye, por defecto de tipo `double`.	`>> A=zeros(3)`
		`>> v=zeros(1,5,"uint8")`

`ones(n,m)`
`ones(n)`
`ones(...,"clase")`
	Como la anterior, pero con valores 1.	`>> A=ones(3)`
		`>> v=ones(1,5,"uint8")`

Mis primeros gráficos

El comando básico para realizar gráficas bidimensionales es:

`plot(x,y)`

donde `x` será el vector de abscisas e `y` será el de ordenadas.

Por ejemplo: teclea en la ventana de comandos

```
>> x=[1,2,3]
>> y=[0.3,2.5,0.9]
>> plot(x,y)
```

Comprobarás que se abre una ventana, con nombre «`Figure 1`», que tiene una botonera con algunas herramientas (desplazamiento, zoom, etc.), puedes moverla, ampliarla, etc. como cualquier otra ventana. Experimenta con los botones...

Ahora vamos a crear varias ventanas. Podemos intentar lo siguiente (teclea en la ventana de comandos):

```
>> x=[1,2,3]
>> y=[0.3,2.5,0.9]
>> plot(x,y)
>> plot([4,5,6],[1,1.5,1])
```

¿Qué ha ocurrido?... que el segundo gráfico ha reemplazado al primero en la ventana de la «`Figure 1`». Para evitar esto hacemos:

```
>> x=[1,2,3]
>> y=[0.3,2.5,0.9]
>> plot(x,y)
>> figure(2)
>> plot([4,5,6],[1,1.5,1])
```

En los ejemplos que siguen repetiremos con frecuencia algunas órdenes en la ventana de comandos: si utilizas la flecha hacia arriba del teclado, en esta ventana, podrás ir repitiendo órdenes y editarlas, si quieres modificarlas.

Algunas órdenes interesantes para el manejo de distintos gráficos

Cierra la ventanas gráficas creadas antes. A continuación teclea en la ventana de comandos las órdenes indicadas en lo que sigue, en el orden indicado.

`figure(n)` Crea una ventana gráfica con nombre «`Figure n`». Si ya existe esta ventana entonces la declara como activa (tiene el foco): es decir, los comandos gráficos siguientes le afectarán a dicha ventana.

```
>> figure(1)
>> plot([1,2,3],[3,4,5])
>> figure(2)
>> plot([1,2,3],[-3,-4,-5])
>> figure(1)
>> plot([3,4,8],[-2,1,-5])
```

¿En qué ventana se ha dibujado la última gráfica?

`clf(n)` Vacía el contenido de la ventana «`Figure n`». Qué es exactamente el contenido y qué desaparece es complicado de explicar ahora: una ventana gráfica es como un contenedor... `clf` elimina ciertas cosas del contenedor, pero no otras y tiene otras sintaxis donde se pueden especificar más detalles de los elementos que se deben eliminar... De momento nos contentamos con esto.

```
>> clf(1)
```

`hold on` Permite añadir elementos gráficos a la ventana gráfica que tiene el foco, sin eliminar los que ya existen.

```
>> figure(2)%%Por si no tiene el foco en estos momentos, lo ponemos en la fig. 2
>> hold on
>> plot([9,10,11,12],[1,2,1,2])
>> hold on
>> plot([3,4,8],[-2,1,-5])
```

`close(n)` Cierra la ventana «`Figure n`».

```
>> clf(1)
```

`close all` Cierra todas las ventanas «Figure...»

Elementos gráficos adicionales

Las posibilidades gráficas en Octave son bastante completas. Aquí vamos a describir sólo algunas de ellas.

En primer lugar, y **muy importante**:

- Podemos incluir varias gráficas simultáneamente en un mismo comando `plot`:

  ```
  >> close all%% Para empezar de nuevo
  >> x1=[1,2,3]
  >> y1=[0.1,2.5,-1.2]
  >> x2=[1,2,3,4]
  >> y2=[-0.1,-2.5,1.2,-3.4]
  >> plot(x1,y1,x2,y2)
  ```

- Octave escala automáticamente el eje de ordenadas para que se ajuste al tamaño de la gráfica (también el de abscisas, si por ejemplo se añade un segundo gráfico a una ventana ya existente). Podemos fijar los límites de los intervalos de abscisas y ordenadas, con las órdenes `xlim([xmin,xmax])`, `ylim[ymin,ymax]`. Por ejemplo, siguiendo con las órdenes ya tecleadas en el punto anterior:

  ```
  >> xlim([-3,2])
  >> ylim[-2,0]
  ```

  Observa lo que ocurre, y utiliza el botón de desplazamiento de la ventana gráfica para comprobar que las gráficas «siguen ahí». Estas órdenes sólo se ocupan de determinar la parte visible del plano... los valores calculados, en abscisas, son los indicados en las variables de `plot`.

Antes de introducir otros elementos de control de gráficos vamos a dibujar algunas gráficas menos triviales que las de los ejemplos anteriores.

Por ejemplo, para dibujar la función $\mathrm{sen}(x)$ en una malla de 200 puntos en el intervalo $[-\pi, \pi]$:

```
>> x=linspace(-pi,pi,200)
>> plot(x,sin(x))%% observa el intervalo de abscisas en el gráfico
>> xlim([-pi,pi])%% ahora el intervalo de abscisas ha cambiado
```

Otro ejemplo, definiendo lo que en Octave se denomina «funciones anónimas» (de las que daremos más detalles en otro momento), que se definen mediante la sintaxis `nombref=@(x,y,...) expresion`:

```
>> mifun=@(x) 2*x./(1+x.^2);
>> x=linspace(-3,3,200);
>> plot(x,mifun(x))
```

Incluyendo puntos en un gráfico

Se pueden incluir puntos en un gráfico pasando a `plot` dos vectores: un vector con las abscisas de los puntos que se quieren dibujar, y un vector con sus ordenadas; además le daremos una indicación de cómo dibujar un punto, mediante el símbolo adecuado, por ejemplo `'*'`, `'o'`:

```
>> mifun=@(x) 2*x./(1+x.^2);
>> x=linspace(-3,3,10);
>> plot(x,mifun(x))
>> hold on
>> plot(x,mifun(x),'*',[0],[0],'o')
```

Colores y leyendas

Podemos controlar muchos otros elementos, como los colores con los que se dibujan las gráficas o los puntos, el estilo de las líneas y los puntos, así como la inclusión de leyendas, títulos de la ventana, etc. He aquí algunos ejemplos.

La orden `plot` tiene numerosas variantes en cuanto a las variables que espera. Una de ellas incluye el «formato», una cadena de caracteres que controla cómo se muestran cada elemento gráfico.

El «formato» es una cadena de caracteres compuesta por cuatro partes, todas ellas optativas. Estas partes son: estilo de la línea, estilo de puntos (marcadores), color y leyenda. Atención: la leyenda, si se incluye debe encerrarse entre signos de punto y coma, es decir, debe ser de la forma `;leyenda;`.

Veamos un ejemplo rápido:

```
>> close all
>> x=linspace(-3,3,200)
>> plot(x,mifun(x),'g--;Mi función;')
>> hold on
>> plot(x,mifun(x),'m*;Puntos;',[0,-3,3],[0,0,0],'o')
```

En el primer caso hemos especificado en el formato el color verde (`g`, de «green>), un estilo de línea discontinnua (`--`), y la leyenda (`Mi función`). En el segundo caso se dibujan sólo puntos (debido al formato especificado con `*`), en color magenta (`m`) y una nueva leyenda para este elemento gráfico adicional.

A continuación tenéis una lista de valores admitidos en el formato:

Estilos de línea Posibles valores: `-` (línea continua, valor por defecto), `--` línea discontinua, `:` línea de puntos, `-.` discontinua compuesta de raya y punto.

Colores Posibles valores `b`, `r`, `g`, `b`, `y`, `m`, `c`, `w`.

Marcas de puntos Entre otros: `+`, `o`, `*`, `.`, `x`, `s`, etc.

Una sintaxis alternativa, y que ofrece más posibilidades es

```
plot(x,y, prop1,valor1,prop2,valor2,...)
```

donde `prop1`, `prop2` representan el nombre de distintas propiedades, seguidas cada una de su valor. Entre las propiedades:

`linestyle`, `linewidth`, `color`, `marker`, `markersize`, `markeredgecolor`, `markerfacecolor`,...

El nombre de la propiedad y los valores que no sean numéricos deben escribirse entre apóstrofos o comillas (como todas las cadenas). Un ejemplo con pequeñas variaciones sobre el anterior:

```
>> close all
>> x=linspace(-3,3,200)
>> plot(x,mifun(x),'color','c','linestyle','-.','linewidth',2)
>> legend('Mi función')
>> hold on
>> plot(x,mifun(x),'marker','*',[0,-3,3],[0,0,0],'marker','o','markerfacecolor','y')
```

Observa la orden `legend` haz algún intento de insertar dos leyendas con esta orden.

Para más detalles, puedes consultar los siguientes enlaces sobre Octave:
https://octave.org/doc/v4.0.0/Two_002dDimensional-Plots.html
https://octave.org/doc/v4.0.0/Two_002ddimensional-Function-Plotting.html
Aunque no es exactamente la misma sintaxis, información sobre Matlab, con ejemplos, en:
https://es.mathworks.com/help/matlab/ref/plot.html
https://es.mathworks.com/help/matlab/ref/fplot.html

Ejercicios para la práctica 1

1. Este primer ejercicio consiste en dibujar algunas gráficas sencillas, para familiarizarnos con los elementos gráficos. Debemos escribir el código en un archivo denominado `Ejercicio1.m`. Como se van añadiendo elementos apartado a apartado, utiliza `close all`, desde la ventana de comandos, para cerrar la ventana gráfica entre apartado y apartado, antes de volver a ejecutar el (también puedes cerrar directamente la ventana con el ratón).

   a) Dibuja en una ventana la gráfica de la función $f(x) = \pi + \sqrt{1+x^2}$ en el intervalo $[-3, 3]$, utilizando 179 puntos. Recuerda que en un script podemos definir una función con el código:

   ```
   f=@(x) pi+sqrt(1+x.^2);
   ```

   Aunque de forma alternativa podemos definirla, dentro del propio archivo script, mediante el código

   ```
   function ret=f(x)
   ret=pi+sqrt(1+x.^2)
   endfunction
   ```

   Aunque en este ejemplo no se ve la necesidad de la segunda opción, es conveniente recordar que la primera sintaxis sólo permite una expresión tras la parte de código `@(x)`. En ambas formas, la definición de la función debe preceder en el código a cualquier llamada a la misma.

   b) Cambia el color de la gráfica y el grosor de la línea (el valor del grosor por defecto es `0.5`). Añade una leyenda.

   c) Añade a la misma ventana la gráfica de la función $g(x) = \text{sen}(x)$ sobre el mismo intervalo. Elige el color de esta nueva gráfica y fija un estilo de línea discontinuo. Incluye también una leyenda.

   d) Finalmente, añade en la misma ventana la gráfica de $h = f + g$.

   e) Añade la orden `title('Mi primer gráfico')`. ¿Qué efecto produce?

   f) Conviene terminar, dentro del archivo `.m`, el código de cada gráfico con `hold off`, para que los parámetros gráficos se reinicien si volvemos a ejecutar el programa... ¡Recuérdalo!

2. Vamos a definir una función a trozos en Octave y a representar su gráfica.

   La principal dificultad para definir una función a trozos en Octave es que se espera que el argumento, `x`, de una función, `f(x)`, sea vectorial, por ejemplo para poder dibujar su gráfica con `plot(x,f(x))`, siendo `x` el vector de las abscisas. Puesto que para poder definir la función por trozos debemos conocer en qué intervalo o región está la variable, debemos realizar comparaciones... ¿Y cómo se comparan vectores?

   a) Escribe en la ventana de comandos

   ```
   >> x=[1,-1,2,-2]
   >> x<0
   ```

   ¿Cómo explicas el resultado?

   b) ¿Qué ocurre si utilizamos la condición `x<0` para tomar una decisión? Escribe los siguientes ejemplos:

   ```
   >> x=[1,-1,2,-2]
   >> if x<0 disp('negativo') else disp('positivo') endif
   >> x=[-1,1,-2,2]
   >> if x<0 disp('negativo') else disp('positivo') endif
   >> x=[1,2,3,4]
   >> if x<0 disp('negativo') else disp('positivo') endif
   >> x=[-1,-2,-3,-4]
   >> if x<0 disp('negativo') else disp('positivo') endif
   ```

   ¿La conclusión es...?

Tenemos dos formas de aproximarnos al problema

a) Manejar las funciones características simuladas en Octave. Recuerda que si A es un conjunto, digamos en $\mathbb{R}$, la función característica de A es:

$$\chi_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases}$$

Así, por ejemplo, si $A = (-\infty, 0)$ y $B = [0, +\infty)$ la función $f(x) := -x\chi_A(x) + x\chi_B(x)$ coincide con $|x|$.

Esto es fácil de escribir en Octave: `f=@(x) -x.*(x<0)+x.*(x>=0)`. Por ejemplo, escribe

```
>> x=[1,-1,2,-2];
>> f=@(x) -x.*(x<0)+x.*(x>=0);
>> f(x)
```

¿Obtienes el resultado buscado? Haz otras pruebas para confirmarlo...

b) La segunda opción para resolver este problema es recorrer las componentes del vector `x` en un bucle y realizar la comparación componente a componente. Por ejemplo, escribe:

```
>> function ret=f(x)
     for i=1:length(x)
      if (x(i)<0)
       ret(i)=-x(i);
      else
       ret(i)=x(i);
      endif
     endfor
   endfunction
>> f([-1,2,-3,-10])
```

c) Terminamos el ejercicio escribiendo un archivo `Ejercicio2.m` donde vamos a representar en dos ventanas distintas las gráficas de las funciones:

$$f(x) = \begin{cases} -1 & \text{si } x \leq -2 \\ (x+2)^2 - 1 & \text{si } -2 < x < 2 \\ 15 & \text{si } x \geq 2 \end{cases} \qquad g(x) = \begin{cases} x\,\text{sen}(\frac{1}{x}) & \text{si } x < 0 \\ 0 & \text{si } x = 0 \\ x^2\,\text{sen}(\frac{1}{x}) & \text{si } x > 0 \end{cases}$$

Dibuja la gráfica de f en el intervalo $[-5, 5]$ y la de g en el intervalo $[-0.25, 0.25]$.

Finalmente añade en la segunda ventana las gráficas de las funciones «moduladoras» de g:

$$m_1(x) = \begin{cases} x & \text{si } x < 0 \\ -x^2 & \text{si } x \geq 0 \end{cases} \qquad m_2(x) = \begin{cases} -x & \text{si } x < 0 \\ x^2 & \text{si } x \geq 0 \end{cases}$$

En cada uno de los casos puedes definir las funciones de forma anónima (con `f=@(x)...`) o mediante un bucle, componente a componente.

Para realizar un condicional con la conjunción de dos condiciones escribe:

```
if ((cond1) & (cond2))... endif
```

3. El método de aproximación de la derivada más elemental, que desarrollaremos en el capítulo 3, es el siguiente: si f es una función de clase $\mathcal{C}^2$ podemos escribir

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0 + \theta h)}{2!}h^2$$

siendo $\theta \in (0, 1)$ un valor que depende de x_0 y h. Esto significa que podríamos aproximar el valor de la derivada $f'(x_0)$ mediante

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

y el error cometido estaría acotado:

$$\left|f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}\right| \leq \frac{|f''(x_0+\theta h)|}{2!}|h|$$

Si $|h|$ es pequeño incluso podríamos aproximar la cota del error por $\frac{|f''(x_0)|}{2!}|h|$.

Así podemos pensar que cuanto menor sea $|h|$ mejor será la aproximación del cociente a $f'(x_0)$. Vamos a ver que esto no es así: la razón es que, para valores muy pequeños de $|h|$, los errores de redondeo, y concretamente el error de cancelación que se produce en el numerador del cociente, provoca un efecto demoledor en la aproximación.

Queremos aproximar el valor de la derivada de $f(x) = \operatorname{sen}(x)$ en el punto $x_0 = 1.2$. Para ello, calcularemos el cociente

$$\frac{\operatorname{sen}(1.2+h)-\operatorname{sen}(1.2)}{h} \tag{1}$$

para distintos valores de h.

Puesto que con Octave podemos calcular en precisión doble el valor de la derivada que buscamos, $\cos(1.2) = 0.362\,357\,754\,476\,674$ (que tomamos como valor exacto), podremos estimar el error en las distintas aproximaciones. Según los comentarios anteriores, el error absoluto debería ser de la forma Ch, para algún valor de C (que dependerá de h), es decir, es del orden de h, $O(h)$. Incluso hemos comentado que el error absoluto podría aproximarse por $\frac{|f''(x_0)|}{2!}|h| \approx 0.466|h|$.

a) Calcula para valores de $h = 10^{-k}$, con $k = 1, 2, \ldots, 7$, el cociente (1) y el error absoluto respecto al valor «real» $\cos(1.2)$ que proporciona Octave. Escribe los valores de salida en forma de tabla ordenada, añadiendo una columna que sea el valor de $\frac{|f''(x_0)|}{2!}|h| = \frac{|-\operatorname{sen}(1.2)|}{2}|h|$. Comprueba que los resultados coinciden con lo que predecimos.

b) Ahora añadimos a la tabla de salida los mismos cálculos para valores de $h = 10^{-k}$, para $k = 8, \ldots, 16$. Observa el resultado y comprueba cómo los errores comienzan a aumentar conforme h disminuye, llegando a valores del orden de 0.1.

c) Ahora vamos a comprobar visualmente cómo evoluciona el error. Para ello realizamos un gráfico con escala logarítmica tanto en las abscisas como en las ordenadas. Para realizar este tipo de gráficos Octave dispone del comando

`loglog(x,y,...)`

que sigue una sintaxis idéntica a la de `plot`, con los posibles argumentos de forma idéntica.

Realiza entonces un gráfico con escalas logarítmicas que muestre las gráficas, como funciones de h, de los errores:

$$e_1(h) := \left|\frac{\operatorname{sen}(1.2+h)-\operatorname{sen}(1.2)}{h} - \cos(1.2)\right| \quad \text{y} \quad e_2(h) = \frac{\operatorname{sen}(1.2)}{2}h$$

Añade puntos sobre la gráfica en cada una de las coordenadas calculadas. Puedes realizar el gráfico utilizando las órdenes:

```
i=-20:-0.5:0;
h=10.^i;%%Valores de h variando entre 10^(-20) y 1
e1=...
loglog(h,e1...)
```

Finalmente, para realizar en casa (si no tienes tiempo durante la sesión de prácticas): representa en una ventana gráfica las funciones $\cos(x)$ y $\varphi_h(x)$, para distintos valores de $h = 10^{-1}, 10^{-3}, \ldots, 10^{-15}$, ambas en el intervalo $[0, \pi]$. La función $\varphi_h(x)$ es el valor aproximado a la derivada de $\operatorname{sen}(x)$ mediante el cociente anterior, es decir:

$$\varphi_h(x) = \frac{\operatorname{sen}(x+h)-\operatorname{sen} x}{h}.$$