



UNIVERSIDAD
DE GRANADA

**Departamento de
Matemática Aplicada**

Prácticas con ordenador: OCTAVE

Sesión 1. Primeros pasos en cálculo numérico

Ingeniería Civil - Administración y Dirección de Empresas

3 de marzo de 2023

1. Algunas direcciones útiles (con información sobre Octave)
2. Errores de redondeo: resultados inesperados
3. Propagación de errores: procesos iterativos inestables
4. Error por cancelación: cálculo de límites
5. Primeros pasos de programación
 - 5.1 Los bucles “for”, “while” y “do ...until”
 - 5.2 La sentencia condicional “if” (“else” y “elseif”)
6. Primeras gráficas
 - 6.1 Ejemplos (ejercicios)
7. El método de bisección para la resolución de ecuaciones no lineales

1. Algunas direcciones útiles (con información sobre Octave)

- ▶ A.M. Delgado, J.J. Nieto, A.M. Robles y O. Sánchez. *Métodos numéricos básicos con Octave*. Editorial Técnica Avicam, Granada, 2016.
<https://github.com/oscarsanchezromero/Calculo-Cientifico-Octave>
- ▶ A. Quarteroni, F. Saleri. “Cálculo científico con MATLAB y Octave”. Springer, 2006. (¿Nivel avanzado?)
- ▶ S.H. Carbonetto. “Tutorial de Octave”. (Gráficos: Cuadro 1, página 13)
https://eva.fing.edu.uy/pluginfile.php/176345/mod_resource/content/0/Tutorial_Basico-Octave_2010_uba.pdf
- ▶ F.J. Pena. “Introducción al Octave”. (General)
<https://www.slideshare.net/franpenabra/introccpresentacion-octave>
- ▶ Comandos básicos de Octave (3.0). (Para consulta rápida)
<https://www.educa2.madrid.org/web/educamadrid/principal/files/34b0304e-7fce-4b92-875e-b0c7711e9926/RECURSOS/CURSOS/CIENCIAS/MATEMATICA/GEOMETRIA/5.1.pdf?t=1352405767458>
- ▶ Instrucciones para realizar gráficos. (Ejemplos útiles)
<https://octave.sourceforge.net/octave/function/text.html>

Observación

- ▶ Cuando hacemos cuentas con un ordenador, algunas propiedades de las operaciones con números pueden dejar de ser válidas.

Ejemplo

Consideremos $a = 10^{-9}$.

- ▶ Calcula $(1 + a) - 1$, $1 + (a - 1)$ y $1 + a - 1$.
- ▶ Calcula $\frac{1+a}{a} - \frac{1}{a}$ y $\frac{(1+a)-1}{a}$.

Comenta los resultados obtenidos tras realizar los cálculos con la opción “format short” y con la opción “format long”.

Observación

- ▶ Debemos tener en cuenta que distintos ordenadores (con igual programa y versión) pueden dar distintos resultados en las operaciones anteriores.
- ▶ Esto se debe a diversos factores (base de trabajo, número de cifras empleadas al operar, redondeo y/o truncamiento usados, etcétera) que, provocando pequeñas diferencias, pueden dar lugar a grandes errores.

Ejemplo

- ▶ Calcula $1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2$,
 $1 - (0.2 + 0.2) - 0.2 - 0.2 - 0.2$ y $1 - (0.2 + 0.2) - (0.2 + 0.2 + 0.2)$.
- ▶ Calcula $\sin(\pi)$, $\cos(\pi)$, $\sin(\frac{\pi}{2})$ y $\cos(\frac{\pi}{2})$.

- Consideremos el siguiente *script*.

```
x=1;  
y=1./3;  
for i=1:20  
    z=10.*y./3-x;  
    t1=1./3.^(i+1);  
    t2=(1./3).^(i+1);  
    disp([z,t1,t2,z-t1,t1-t2]);  
    x=y;  
    y=z;  
end
```

- Teóricamente, en cada paso, los valores obtenidos para t_1 , t_2 , z han de ser iguales entre sí.

Observación

- ▶ Cuando sólo conocemos una función por los valores que toma en su dominio, una manera de determinar su derivada es utilizando la *derivación numérica*.
- ▶ La derivación numérica consiste en tomar como derivada los valores dados por el *cociente incremental* cuando el denominador es “pequeño”. Es decir,

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

- ▶ Sin embargo, la división por valores pequeños y la diferencia de valores próximos produce resultados no deseables.

Ejemplo

- Examinemos la situación descrita con el siguiente *script*.

```
x=pi;  
for i=1:30  
    h=10.^(-i);  
    der=(sin(x+h)-sin(x))./h;  
    disp([i,der]);  
end
```

- Repite el *script* con $x=0$ y con $x=\pi/2$.
- Repite el *script* con la función coseno y los valores $x=0$, $x=\pi/2$, $x=\pi$.

Por medio de un ejemplo veremos como funcionan una serie de sentencias muy útiles a la hora de realizar algoritmos.

Ejemplo

Sumemos los cuadrados de los cien primeros números enteros positivos.

- Un *script* que usa el bucle “for”.

```
suma=0;
for k=1:100
    suma=suma+k.^2;
end
disp(suma)
```

Ejemplo (continuación)

- Un *script* que usa el bucle “while”.

```
suma=0;  
k=1;  
while k<=100  
    suma=suma+k.^2;  
    k=k+1;  
end  
disp(suma)
```

Ejemplo (continuación)

- Un *script* que usa el bucle “do ...until”.

```
suma=0;  
k=1;  
do  
    suma=suma+k.^2;  
    k=k+1;  
until k>100  
disp(suma)
```

- ¿Cómo modificarías los tres *scripts* anteriores para sumar los cuadrados de los cien primeros números enteros positivos pares?

Ejemplo

Vamos a construir tres funciones. ¿Qué hace cada una?

- Un *script* que sólo emplea el comando “if”.

```
1;  
function y=f1(x)  
    if x>0  
        y=x.^2-5;  
    end  
end
```

Ejemplo (continuación)

- Un *script* que emplea el comando “if” + “else”.

```
1;  
function y=f2(x)  
    if x>0  
        y=x.^2-5;  
    else  
        y=x-5;  
    end  
end
```

Ejemplo (continuación)

- Un *script* que emplea el comando “if” + “else” + “elseif”.

```
1;  
function y=f3(x)  
for i=1:length(x)  
    if x(i)>0  
        y(i)=x(i).^2+x(i);  
    elseif x(i)<-2  
        y(i)=x(i)+2;  
    else  
        y(i)=0;  
    end  
end  
end
```

Observación

- ▶ En el tercer *script* del ejemplo, para $f3$ hemos usado argumentos vectoriales. Esto nos permitirá hacer la gráfica correspondiente directamente.
- ▶ Por contra, tal como se han construido $f1$ y $f2$, no es posible usar directamente la orden `plot` para representarlas (salvo que hagamos lo que se muestra en la subsección 6.1).

Observación

- ▶ Cuando se dibujan gráficas con Octave, en realidad lo que se pinta es un conjunto de pares de puntos.
- ▶ Puede parecer que no es así. En efecto, al ejecutar la orden

```
fplot('sin(x)', [0,pi]);
```

aparece la gráfica del seno. En este caso, el programa decide los puntos que pinta.

Ejemplo

- El siguiente *script* representa qué hace Octave.

```
1;  
function y=f(x)  
    y=1./(1+25*x.^2);  
end  
z=linspace(-1,1,100);  
plot(z,f(z))
```

- Con ayuda de las referencias (en particular, la sección 6 del tutorial de S.H. Carbonetto), haz que la gráfica del *script* anterior cambie de color, anchura, tipo de marcador, etcétera.
(Cuidado con las opciones que sólo son válidas para Matlab).

Ejemplo

- Veamos cómo representar la gráfica de una función a trozos sin emplear argumentos vectoriales

```
1;  
function y=f(x)  
    if x>2  
        y=x.^2+2;  
    else  
        y=x.^3-2;  
    end  
end  
function y=g(x)  
    y=0;  
end  
z=linspace(-8,8,100);  
for i=1:length(z)  
    fz(i)=f(z(i));  
    gz(i)=g(z(i));  
end  
plot(z,fz)  
hold on  
plot(z,gz)  
hold off
```

Ejemplo (continuación)

- ▶ La función $g(x)$ se ha introducido para representar el eje de abscisas.
- ▶ Usando `plot(gz,z)` podríamos representar el eje de ordenadas.

- ▶ Cuando hicimos uso de las sentencias `if`, `else` y `elseif`, se comentó que hay problemas para dibujar las gráficas de funciones definidas por partes.
- ▶ Con los siguientes ejercicios se pretende eliminar tales problemas mediante varias alternativas.

- 1) Intenta dibujar las gráficas de f_1 y f_2 . Como posibles vías, intenta
 - ▶ pintar cada trozo por separado y luego agruparlos;
 - ▶ crear listas para cada trozo y pintarlos en una única gráfica.

(Pueden ser útiles los comandos `hold on` y `hold off`, tal como se han usado en el ejemplo de la página 18/23).

- 2) Dibuja la gráfica de f_3 .
- 3) Modifica los *scripts* de f_1 y f_2 para poder hacer su gráfica directamente (es decir, tal como se ha hecho con f_3).

Observación

- ▶ Para aproximar los ceros de una función (continua) podemos emplear el método de bisección (consecuencia del Teorema de los ceros de Bolzano).

Ejemplo

- ▶ Un primer esbozo de la implementación de este método es el siguiente *script*.

```
1;  
function y=f(x) y=x.^2-3; end  
a=0; b=2;  
for i=1:20  
    c=(a+b)./2;  
    if f(a).*f(c)<0  
        b=c;  
    else a=c;  
    end;  
    printf("la iteración %i es igual a %.8f \n",i,c);  
end
```

- ▶ Para entender el funcionamiento de %i, %.8f, \n y comando similares, puedes consultar el archivo "comandostablas.m".
- ▶ Se puede emplear "fprintf" en lugar de "printf" (la diferencia entre estas dos órdenes queda fuera del alcance de esta asignatura).

Observación

- Hay que tener cuidado al definir las condiciones en los bucles.

Ejemplo

- En el siguiente *script* hay un error. ¿Dónde?

```
1;  
function y=f(x) y=7.*x-1.4; end  
a=0; b=0.8;  
for i=1:20  
    c=(a+b)./2;  
    if f(c)==0  
        printf("hemos encontrado la solución exacta");  
    elseif f(a).*f(c)<0  
        b=c;  
    else a=c;  
    end;  
    printf("la iteración %i es igual a %.8f \n",i,c);  
end
```

- ¿Cómo arreglarías el *script*?
- Realiza las mejoras que consideres oportunas en el anterior *script*.



Departamento de Matemática Aplicada. Universidad de Granada

Licencia Creative Commons 3.0 España.

<http://creativecommons.org/licenses/by-nc-sa/3.0/es/>