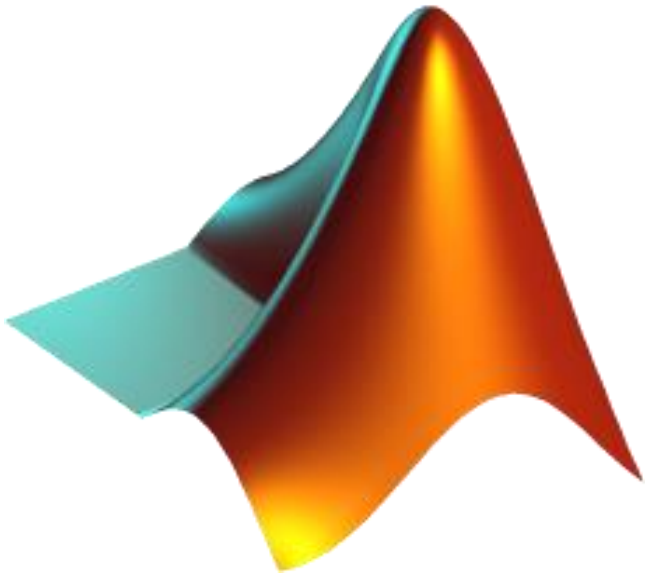
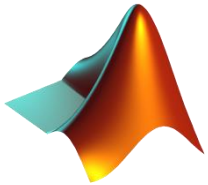
UNIDAD 1.
Introducción y conceptos básicos.

- MATLAB. Interfaz/entorno gráfico (*desktop*).
- Operaciones aritméticas elementales.
- Funciones: `cos`, `tan`, `log`, etc.
- Variables.
- Vectores.
- Representación gráfica básica.
- Ficheros de comandos (*scripts*).
- Derivada numérica.
- Integral numérica.
- Representación gráfica (más utilidades).
- Importar y exportar datos.



"Hay que aprender a hacerse entender por una máquina"



MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices")

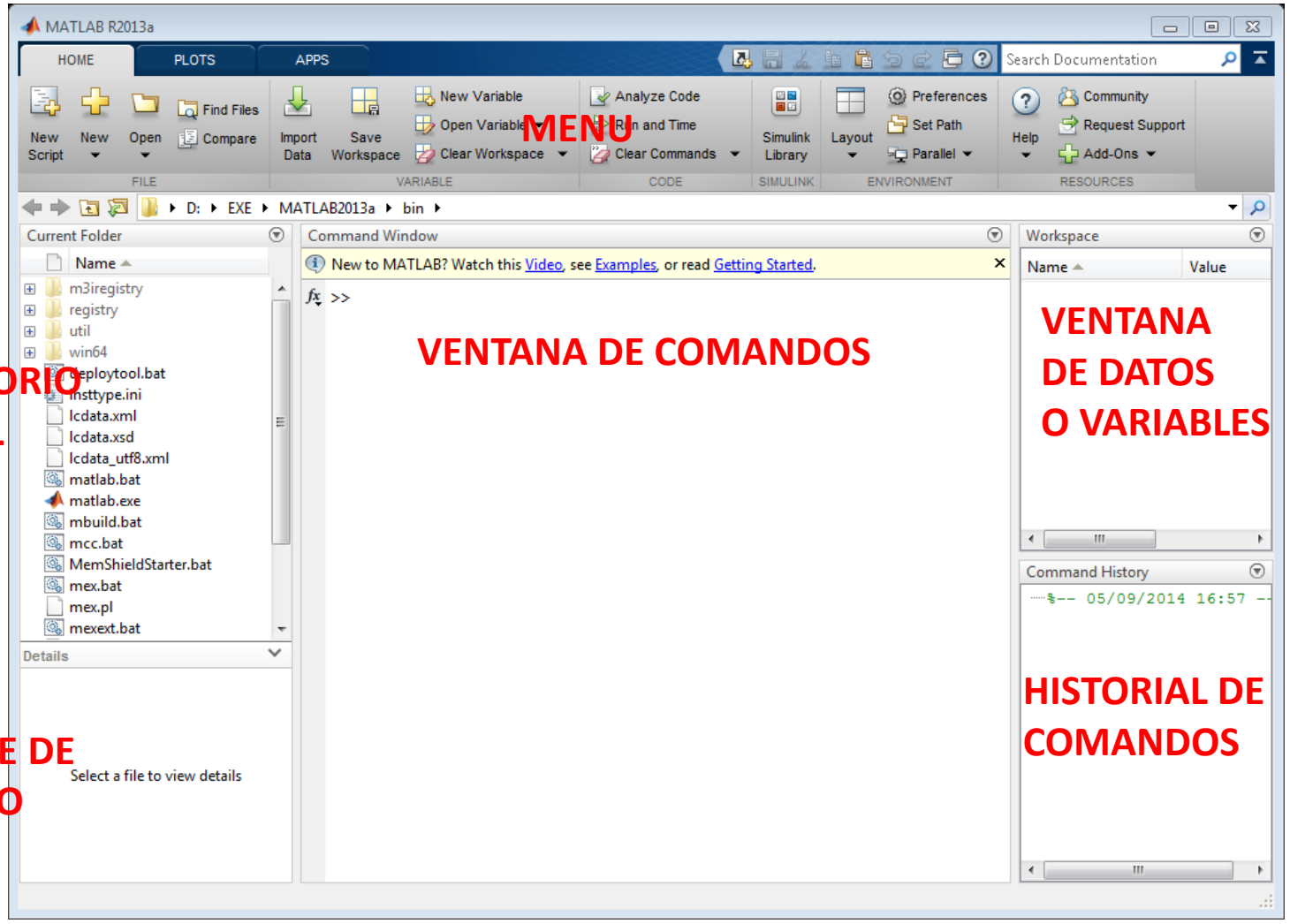
Es una herramienta de software matemático que ofrece un interfaz gráfico con un lenguaje de programación propio (lenguaje M).

Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la operaciones y manipulación de vectores y matrices, la representación de datos y funciones, la implementación de algoritmos (programas), la comunicación con programas en otros lenguajes (C, C++, etc.) y con otros dispositivos hardware.

Alternativas freeware: Octave, SciLab.

MATLAB: **Interfaz/entorno gráfico.**



ATENCIÓN: es configurable por el usuario.

MATLAB: Ventana de comandos

- Todas las operaciones pueden introducirse (mediante texto) y ejecutarse en la línea de comandos (tecla *enter*).
- Distingue entre mayúsculas y minúsculas.
- La línea de comandos está preparada cuando el cursor es >>.
- Signo decimal es punto. La coma tiene otro significado.

Ejemplo:

```
>> 3.54
```
La (última) respuesta/*answer* (`ans`) es 3.54

```
>> -1.42E-10
>> 3,54
```
La respuesta no es la esperada (la coma es un signo especial en MATLAB)

MATLAB: Operadores aritméticos elementales

- Operadores aritméticos elementales: `+, -, *, /, ^` (exponenciación).
- Atención al orden de ejecución de los operadores (`^*/+-`) y a la precedencia de los paréntesis (de dentro a fuera)

Ejemplo:

```
>> (5.66-4.32)^0.34
>> 5.66-(4.32^0.34)
>> 5.66-4.32^0.34
```

Números complejos: compuestos por parte real e imaginaria. Se representa mediante la suma de la parte real y la imaginaria (`i` o `j`).

Ejemplo:

```
>> -1^0.5      ans     -1
>> (-1)^0.5    ans     0.0000 + 1.0000i
```

Funciones con números complejos: `complex real imag abs conj angle`

Obsérvese la utilidad de la ventana del historial de comandos:

- Doble click ejecuta de nuevo el comando
- Click derecho -> borrar historial de comandos (a parte de otras opciones)

- También para repetir comandos anteriores/posteriores pulsar flecha arriba/abajo en la ventana de comandos

MATLAB. Funciones *predefinidas*

El argumento de la función se coloca entre paréntesis.

Trigonométricas y sus inversas (en radianes!)
Ejemplo: `sin(0.4), tan(0.3), asin(0.4),` etc.

Exponenciales y logarítmicas
Ejemplo: `exp(2.3), log(34.5), log10(100),` etc.

Redondeo
Ejemplo: `round(3.5) fix(-2.1) floor(-2.1) ceil(-2.1)`

Con más de un argumento (se separan con comas)
Ejemplo: `max(-6,5) , min(-6,5) , rem(4,3), mod(4,3),` etc

Otras
`sqrt(4.0)` (raíz cuadrada)
`sign(-8.4)` (signo)
`factorial(3)`

Otros cientos…

MATLAB. Mensajes de error.

Cuando el comando no puede ejecutarse correctamente (por error del usuario) aparece en rojo, proporcionando alguna información de la causa.

Ejemplos

```
>> log10
Error using log10
Not enough input arguments.

>> 4^^3
4^^3
Error: Unexpected MATLAB operator.

>> exp(sevilla)
Undefined function or variable 'sevilla'.
```

MATLAB. Ayuda.

```
>> help('log10')
>> lookfor('logarithm')
```

Atención a las comillas simples!

MATLAB. VARIABLES.

Variables predefinidas (de MATLAB): valores numéricos asociados a un nombre, entre otras:

```
>> pi            (3.1415…)
>> i, j, 1i, 1j          sqrt(-1)
>> NaN , Inf             Not a Number, Infinity
>> ans                   último resultado válido
```

Variables definidas por el usuario: valores numéricos asociados a un nombre definido por el usuario. Se definen mediante el signo igual =

Ejemplo: `var_nombre=valor numérico.`

```
>> velocidad_5=4.32
velocidad_5 =
    4.3200
```

Reglas para el nombre (que aclare el contenido de la variable!):

- Empieza (siempre) por una letra.
- Puede contener una combinación de letras (sin acentos), números, _(subrayado).
- No puede contener espacios.
- Distingue (siempre) entre mayúsculas y minúsculas.

Ejemplo: `>> vel_1_x=log(3+sqrt(8))`

Ejemplo: `>> deg2rad=pi/180 ; rad2deg=180/pi ;`

Variables. Detalles

- Puede haber tantas como desee el usuario.
- Contienen el último valor asignado por el usuario (se sobrescriben).
- Se mantienen sólo durante la sesión, aunque pueden guardarse en un fichero antes de salir de MATLAB (más adelante se verá).
- Pueden contener valores reales o complejos, entre otros (más adelante se verá).

El **workspace** y la **ventana de datos y variables**.

- En el *workspace* se almacenan todas las variables definidas por el usuario.
- Se muestran en la ventana del *workspace*.
- No se puede, ni tiene sentido, almacenar valores que no estén asociados a una variable (con su nombre).
- El workspace también almacena la última respuesta válida (`ans`).
- Comando `who`: muestra la lista de variables actuales.

Borrado de variables

Comando `clear var_nombre`

Comando `clear all` (borra todas)

Click derecho en ventana de variables

MATLAB. Otras utilidades.

Utilizar punto y coma al término de un comando inhibe que se muestre el resultado (`ans`).

Comando `clc`: borra la ventana de comandos.

Comando `home`: sitúa el cursor en la última línea de comandos.

Comando `format`: establece el formato con que se muestran los valores numéricos

```
long / short      compact / loose
>> format long
```

Varios comandos en la misma línea se separan mediante coma o punto y coma.

```
>> format long ; pi , format short ; pi
```

Interrumpir ejecución de un comando con la combinación de teclas Ctrl-C

```
>> [1:10000000]
```

Vectores.

MATLAB está especialmente diseñado para operar eficazmente con vectores (o ristras ordenadas de datos) y matrices. No siempre se corresponde con un vector geométrico (XYZ) o uno de álgebra.
Se definen como variables mediante corchetes. Las columnas se separan mediante coma (o espacio) y las filas mediante punto y coma.

```
>> vector_fila_1=[1,3,5]
>> vector_fila_2=[1 3 5]
>> vector_columna_1=[4;5;6]
```

(NOTA: no hace falta espacio tras la coma o el punto y coma)

- El nombre de la variable sigue las reglas estándar.
- Los valores equivalen a un vector de un elemento.
- Pueden contener valores reales o complejos (entre otros, se verá más adelante).

Pueden visualizarse en forma de tabla de datos (doble click en el workspace).

Vectores. Utilidades.

El operador 'dos puntos': genera un vector de números entre dos valores reales, opcionalmente con un cierto incremento que es 1 si no se especifica:

`[principio:incremento:fin]` **`help colon`**

```
>> [3:1:8]
```

`principio:incremento:final` Es lo mismo

```
>> a=4:6                          NOTA: incremento 1
>> dent=[-4:3:19]                 NOTA: atención a si no encaja
>> c_0=[5:-2:-2]                  NOTA: incremento negativo
>> c_1=[0.5:0.1:(1.0+0.2)]        ATENCIÓN a los paréntesis y decimal
>> c_2=[5:-1:6]                   ATENCIÓN al error
```

Para construir vectores formados un número de valores (equiespaciados) se utiliza la función: `linspace(principio,final,número_de_valores)`

```
>> bb=linspace(0,4*pi,1024);      ATENCIÓN al punto y coma
>> z_2=linspace(10.4,0.2,100) ;   ATENCIÓN: incremento negativo
```

Funciones de vectores

Muchas funciones pueden operar *de una sola vez* sobre cada uno de los elementos de un vector

```
>> a=linspace(0,4*pi,1024) ; b=sin(a);
>> plot(x,y);                     Se verá con detalle más adelante
```

Vectores. Operaciones '*elemento a elemento*'.

La operación se lleva a cabo sobre todos y cada uno de los elementos de forma independiente.

Multiplicación o división por un escalar (de todos los elementos del vector)

```
>> x=[0:9] ; xx=5.43*x
>> x=[0:9] ; xx=x/4.56
```

Suma o resta de un escalar (a todos los elementos del vector)

```
>> x=[0:9] ; xx=x-4.34
```

Suma o resta de vectores (deben contener el mismo número de elementos)

```
>> x=[-1:9] ; xx=x+sqrt(x)
```

Multiplicación (.*), división (./) o exponenciación (.^) de vectores 'elemento a elemento'. Nota: tienen que tener el mismo número de elementos y ser del mismo tipo (fila o columna). ATENCIÓN al punto antes del operador: Los operadores `*, /` y `^` aplicados a vectores funcionan de forma completamente diferente.

```
>> x=[0:3] ; y=[-2:1] ; z=x.*y
>> z=x.^2
>> z=[ 2 3].*[5 6 7]          error
>> z=[2,3].*[3;5]             error
```

Vectores. Operaciones vectoriales.

Módulo de un vector con la función `norm(vector_1)`

```
>> a=[3:7] ; norm(a) , sqrt(dot(a,a))
```

Producto escalar de dos vectores con la función `dot(vector_1,vector_2)`. Los vectores deben contener el mismo número de elementos. El resultado es un escalar.

```
>> a=[1:3] ; b=[3:5] ; c=dot(a,b) , c=sum(a.*b)
>> a=[1:3] ; b=[3:5] ; a=acos(dot(a,b)/(norm(a)*norm(b)))
```

Producto vectorial de dos vectores con la función `cross(vector_1,vector_2)`. Los vectores contener el mismo número de elementos. El resultado es un vector.

```
>> a=[1:3] ; b=[3:5] ; c=cross(a,b)
```

Ejercicio: comprobar que `c` es ortogonal tanto al vector `a` como al vector `b`.

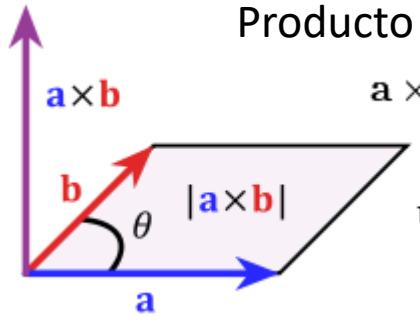
ATENCIÓN: El producto matricial (operador `*` aplicado a vectores) es completamente distinto, se verá más adelante. Recordar que el operador `.*` es una multiplicación elemento a elemento.

Producto escalar

$$\mathbf{A}\cdot\mathbf{B} = \|\mathbf{A}\|\,\|\mathbf{B}\|\cos\theta,$$

$$\mathbf{A}\cdot\mathbf{B} = \sum_{i=1}^{n} A_i B_i = A_1B_1 + A_2B_2 + \cdots + A_nB_n$$

Producto vectorial



$$\mathbf{a}\times\mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\sin\theta\,\mathbf{n}$$

$$\mathbf{u}\times\mathbf{v} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$

Vectores. Acceso a los elementos de un vector.

Se hace empleando el índice (posición dentro del vector) del elemento del vector entre paréntesis.
El primer elemento es el número uno. Válido para vectores fila o columna.

```
>> a=[0:10]  ; a(1) , a(11), a(12)
```

Posible mensaje de error: `Index exceeds matrix dimensions.`

Asignación de un solo elemento de un vector

```
>> a=[0:5] ; a(2)=44 ; a
```

Acceso a *sub-vectores*, i.e., a conjuntos de elementos. Puede hacerse empleando un vector de índices.

```
>> a=[0:5] , b=a([3,5])   % ATENCIÓN: no necesariamente contiguos!
>> a=[0:5] , b=a([3:5])
```

Ejercicio: de un vector `A` de diez elementos, extraer otro vector `B` que contenga los elementos con índice par de `A`.

El último elemento de un vector puede especificarse mediante `end`.

```
>> a=[0:5] ; a(end)
```

Ejercicio: de un vector `A` de diez elementos, extraer otro vector `B` que contenga desde el segundo hasta el penúltimo elemento de `A`.

Variables de texto

Además MATLAB permite crear variables que contiene texto, que se especifica entre comillas simples.

```
>> a='hola mundo' ;
```
NOTA: puede incluir espacios.

NOTA
No es posible crear vectores columna de textos.
```
>> a=['hola'; 'adiós']
```
error

Utilidad para encadenar texto (juntar texto)
```
>> a=['hola',' adiós']
```
Atención al espacio y al acento

No se deben mezclar/operar variables de texto con numéricas.

Para convertir un valor numérico en su texto, el comando `num2str`:
```
>> a=3.14 ; b=num2str(a) ; a , b, a+b
```

Ejercicio: observar la diferencia entre `a=2.5` y `b='2.5'`.

Representación gráfica (básica). Comando 'plot'.

Genera una representación gráfica de una colección de valores en forma de abscisas y ordenadas, que en general se especifican mediante vectores con el mismo número de elementos. ATENCIÓN: el gráfico aparece en otra ventana.

```
plot(vector_abscisas,vector_ordenadas)
```

```
>> a=linspace(0,6*pi,1024); b=sin(a) ; plot(a,b) ;
>> a=linspace(0,6*pi,1024); plot(a,sin(a));
```

NOTA: no existirá la variable `b`

Color (`r, g, b,…`), tipo punto (`*, o, +,…`), líneas entre puntos(`-, --, :,…`)

```
>> plot(a,b,'r*-') ;          (help plot)
```

Etiquetas de los ejes, con las funciones `xlabel` e `ylabel`: (`help xlabel`)

```
>> xlabel('velocidad (m/s)') ;
```

Etiqueta de título, con la función `title`: (`help title`)

```
>> title('nombre de la gráfica') ;
```

Límites de los ejes, con las funciones `xlim` e `ylim`. (`help xlim`)

```
>> xlim([-0.2 4.5]) ;
```

ATENCIÓN: en todos los gráficos que se presenten deben aparecer correctamente etiquetados los ejes: magnitud que se representa y unidades físicas.

Continuación. Representación gráfica (básica). Comando 'plot'.

Caracteres especiales: `\omega, \Omega, \alpha,…`

```
>> xlabel('ángulo \alpha(rad)') ;
```

Texto en posición arbitraria con el comando `text(X,Y,'texto')`

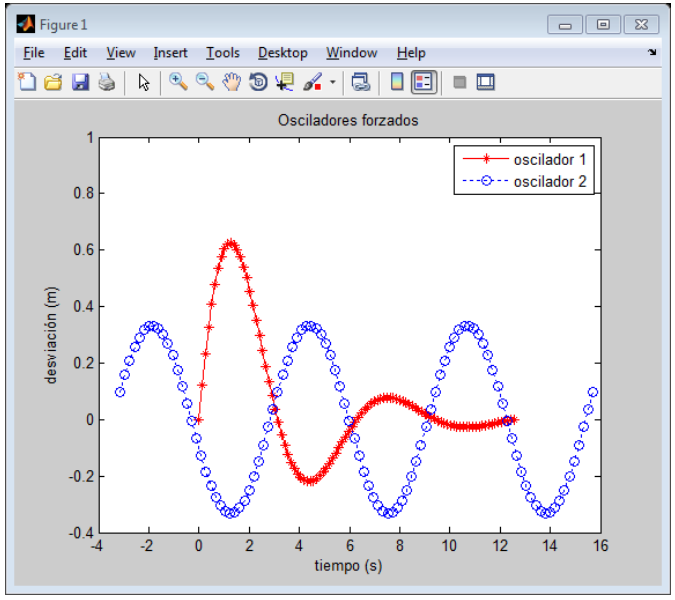
```
>> text(0.1,0.2,'magnitud') ;
```

Representación gráfica de varios *vectores* (`b` frente `a` y `d` frente a `c`):

```
>> plot(a,b,'r*-',c,d,'bo:') ;
```

Leyenda

```
>> legend('oscilador 1','oscilador 2') ;
```



Representación gráfica. Más utilidades.

Útil para representar varios vectores en el mismo gráfico:

Comandos `hold on` y `hold off`: cada comando `plot` sustituye el gráfico anterior, que es equivalente a la instrucción `hold off`.
Si se emplea `hold on` los sucesivos comandos `plot` se añaden al gráfico anterior.

```
x1=linspace(0,4*pi,100) ; y1=sin(x1).^2 ;
plot(x1,y1,'r') ;
y2=sin(x1).^4 ;
hold on ;
plot(x1,y2,'b') ;
hold off ;
```

Comandos:

`grid on (off)` pinta (o no) una cuadrícula sobre la gráfica
`figure(n)` Crea una figura número n (si no existe) o activa la figura n.
`close(n)` Cierra la figura n
`close all` Cierra todas las figuras
`axis equal` los ejes de abscisas y ordenadas tienen el mismo intervalo
`axis square` los ejes de abscisas y ordenadas tienen la misma longitud en pantalla

Representación gráfica. Más utilidades.

Varias gráficas en una misma figura. Comando `subplot(n,m,p)` selecciona el gráfico número `p` de la matriz de gráficos de `n` filas y `m` columnas.

```
x1=linspace(0,4*pi,100) ; y1=sin(x1).^2 ;
subplot(2,3,1) ;
plot(x1,y1,'r') ;
y2=sin(x1).^4 ;
subplot(2,3,5) ;
plot(x1,y2,'b') ;
axis equal ;      % sólo el subplot 5
grid on ; % mostrar retícula
```

EJERCICIO: Construir dos vectores, x e y, que contengan las posiciones iniciales de un conjunto de 100 partículas en forma de espiral, que expresada en forma paramétrica (en metros) es: x(n)=n*sin(n/10) , y=n*cos(n/20), con n=1 a 100.

```
n=[1:1:100] ; % índice de partícula
x=n.*sin(n/10) ; y=n.*cos(n/10) ; % posiciones en espiral
```

Representar gráficamente las partículas mediante círculos rojos y etiquetar la figura y los ejes.

```
plot(x,y,'or') ;
xlabel('posición x (m)') ;
ylabel('posición y (m)') ;
xtitle('posición inicial de las partículas') ;
```

Marcar con un asterisco negro la partícula 20 y hacer que los ejes x e y tengan la misma escala.

```
hold on ;
plot(x(20),y(20),'*b') ; % marcar partícula 20
hold off ;
axis equal ; % ejes iguales
```

Ejercicio: Repetir el ejercicio representando sólo las partículas con n par ;

Ficheros Script o fichero .m

Resulta muy útil escribir un fichero de texto con una secuencia de comandos, que puede modificarse y guardarse en el disco duro o pendrive.

- Abrir editor de texto en el *Menú->New Script.*
- Seleccionar un nombre que aclare su funcionalidad.
- Seleccionar directorio de trabajo (Ventana *Current Folder*).
- Guardar en disco (*Menú->Save*).
- Emplear el carácter % para añadir comentario.
- Ejecutar el Script con *Menú->Run*.

```
% Dibuja coseno de amplitud 4.3
% Rango abscisas = [0:2*pi]

% Borrar todas las variables y ventana de comandos
clear all ; clc ; % otro comentario

% Generar vectores de abscisas y ordenadas
a=linspace(0,2*pi,30) ;
b=4.3*cos(a) ;

% Dibujar
plot(a,b) ;
```

```
% Añadir etiquetas de ejes y título
plot(a,b,'g*') ;
xlabel('tiempo (s)') ;
ylabel('desplazamiento (cm)') ;
title('movimiento oscilatorio') ;
```

ATENCIÓN: los resultados del *script* (`ans`) aparecen en la ventana de comandos, así como los errores.

Ejercicio: Representar gráficamente (200 puntos) la siguiente función f(x) entre *x=0* y *x=100.0*, que representa la velocidad angular omega en radianes por segundo en función del tiempo en segundos. El nombre del *script* debe ser *oscilador_01.m*

$$f(x) = xe^{-x/10}\cos(x)$$

Ejercicio: El número *e* (número de Euler) puede calcularse mediante:

$$F(n) = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

Escribir un *script* llamado *euler_01.m* que calcule y represente gráficamente la función *F(n)* para *n* desde 0 hasta 100, y observar si tiende asintóticamente al número *e*.

Los *scripts* pueden ejecutarse por secciones, que se separan mediante los caracteres %%

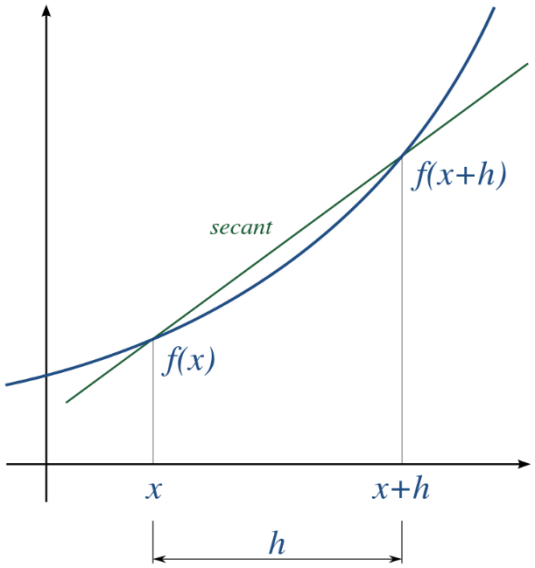
```
%%
% sección 1

%%
% sección 2
```

Para ejecutar sólo una sección, colocar el cursor en la sección y pulsar Shift-Intro, o el icono correspondiente del menú del editor.

Derivada numérica.

Se trata de encontrar la derivada *numérica* de una función arbitraria, cuando por la complejidad de la función sea difícil el cálculo de su derivada analítica.

Puede ser un cálculo útil en el análisis cinemático del movimiento de una partícula, en una, dos o tres dimensiones, así como en multitud de diversas situaciones.

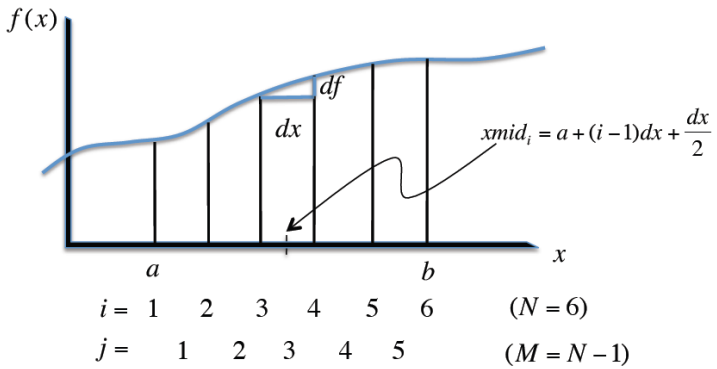


Derivada numérica

Derivada analítica

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Derivada numérica



Dados dos vectores del mismo número de elementos, uno de la variable independiente (*x*) en intervalo [*a,b*] y otro de la variable dependiente (*y*) que es función de *x*, esto es, *y=f(x)*, obtener un vector de la derivada *df(x)/dx*.

Algoritmo:

- Crear un vector *x* en el intervalo [*a,b*], con *N* valores intermedios.
- Crear el vector *y=f(x)* que es función de *x*.
- Calcular el vector $dx=x_{i+1}-x_i$ (que tiene un elemento menos que *x*)
- Calcular el vector $dy=y_{i+1}-y_i$ (que tiene un elemento menos que *x*)
- Calcular el vector *dy_dx=(dy)/(dx)* (elemento a elemento!)

Sea la función $f(x)=x^2+5x$. Obtener su derivada numérica y compararla con la analítica

```
%=====================================
% derivada_01.m
%=====================================

%% Parámetros: intervalo x [a,b] y número de puntos (N)
a=1.5 ; b=15.4 ; N=100 ;

%% Función
x=linspace(a,b,N) ; % variable independiente
y=x.^2+5.*x ; % variable dependiente

%% Derivada numérica
dx=x(2:end)-x(1:end-1) ; % sustituible por diff(x)
dy=y(2:end)-y(1:end-1) ; % sustituible por diff(y)
dy_dx=dy./dx ; % sustituible por diff(y)./diff(x)

%% Derivada analítica
dy_dx_exacta=2.*x+5 ;

%% Representación gráfica
plot(x(1:end-1),dy_dx,'r*', x(1:end-1),dy_dx_exacta,'b-') ;
xlabel('x') ; ylabel('f(x)=2x+5') ;
legend('numérica', 'analítica') ;
```

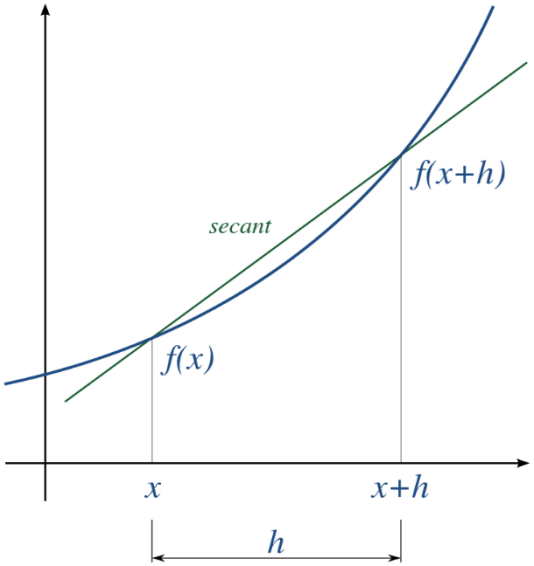
A continuación se muestra un algoritmo que permite obtener la derivada de manera que el número de valores obtenidos no difiera del de la función sobre la que se opera.

El análisis puede encontrarse en la siguiente páginas web:

https://es.wikipedia.org/wiki/Derivación_numérica

Derivación numérica

El método consiste en emplear la diferencia hacia adelante para el primer valor de la derivada, la diferencia hacia atrás para el último valor y las diferencias centrales para el resto de valores intermedios.



Si la función es f(x), su derivada numérica dfdx viene dada por:

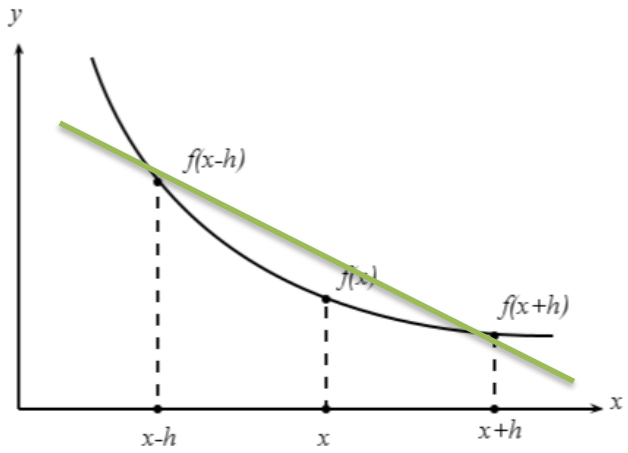
dfdx(1)=(f(2)-f(1))/(x(2)-x(1))
dfdx(end)=(f(end)-f(end-1))/(x(end)-x(end-1))
dfdx(2:end-1)=(f(3:end)-f(1:end-2))./((x(3:end)-x(1:end-2)))

Diferencias hacia adelante:

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

Diferencias hacia atrás:

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



Diferencias centrales:

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$$

Integral numérica.

Se trata de encontrar la integral numérica (definida) de una función arbitraria, cuando por la complejidad de la función sea difícil el cálculo de su integral analítica.

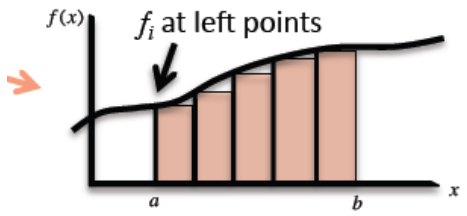
Puede ser un cálculo útil en el análisis cinemático del movimiento de una partícula, en una, dos o tres dimensiones, así como en multitud de diversas situaciones.

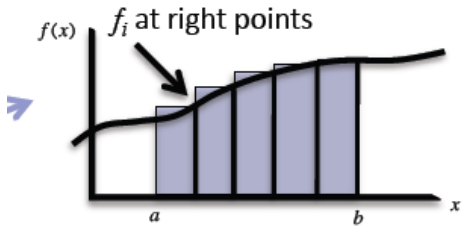
Integrales numéricas

Dada una integral definida de una función f(x) entre los límites del intervalo [*a,b*], la interpretación geométrica es el área bajo la curva (considerando el signo).

La integración numérica más sencilla se corresponde con el cálculo de esa área, discretizando la función a intervalos regulares y *estrechos*. El resultado es la suma del área de todos los rectángulos (véase figura).

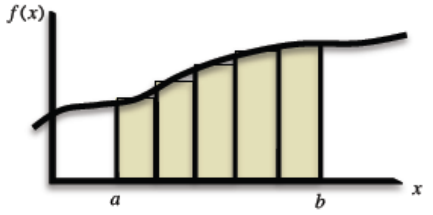
$$I = \int_a^b f(x)dx$$





El área de cada rectángulo es:

$$area = (x_{right} - x_{left})\,\frac{(f_{right} + f_{left})}{2}$$



Integral definida numérica. Algoritmo. Función `sum`.

```
%================================================
% integral_01.m : integra la función coseno
%================================================

%% Define intervalo y número de puntos
N=1000 ; a=0.1 ; b=0.5 ; % ángulo en radianes

%% Calcula variable independiente y función
x=linspace(a,b,N) ;
y=cos(x) ;

%% Calcula incrementos dx y áreas por la izquierda
areas=diff(x).*(y(1:end-1)+y(2:end))/2 ;

%% Área total bajo la curva
integral=sum(areas) ;

%% Muestra resultado y compara con el resultado analítico
integral
integral/(sin(b)-sin(a))
```

La integral incremental o cumulativa

Para aclarar este concepto podemos tomar un ejemplo físico de la cinemática, en el que se calcula la posición de la partícula *x(t)* a partir de su posición inicial y el cálculo de la integral de la velocidad *v(t)*.

$$\frac{dx(t)}{dt} = v(t)$$

$$\int_{t_0}^{t} dx = \int_{t_0}^{t} v(t')dt'$$

$$x(t) - x(t_0) = \int_{t_0}^{t} v(t')dt'$$

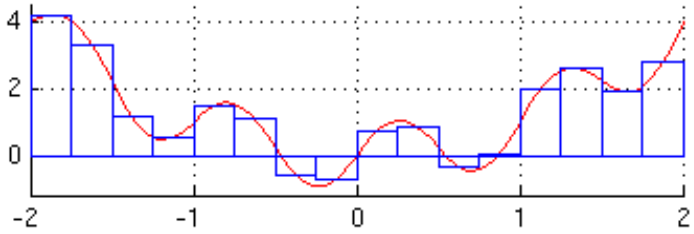
$$x(t) = x(t_0) + \int_{t_0}^{t} v(t')dt'$$

Esta última integral es precisamente la *incremental* o *cumulativa*, esto es, los valores la integral definida para todos los instantes de tiempo entre t_0 y t.

Integral numérica cumulativa o incremental

Sirve para encontrar la integral definida de una función entre un límite inferior fijo (*a*) y un límite superior que va incrementándose hasta un valor *b*. Esto es, se pretende obtener la integral definida desde *a* hasta un conjunto de valores entre *a* y *b*.
Para ello puede utilizarse la función `cumsum` que devuelve un vector.

```
>> integral_cum=cumsum(areas) ;
```



A continuación se muestra un algoritmo que permite obtener la integral numérica de manera que el número de valores obtenidos no difiera del de la función sobre la que se opera.

El análisis puede encontrarse en la siguiente páginas web:

https://es.wikipedia.org/wiki/Integración_numérica

Integración numérica

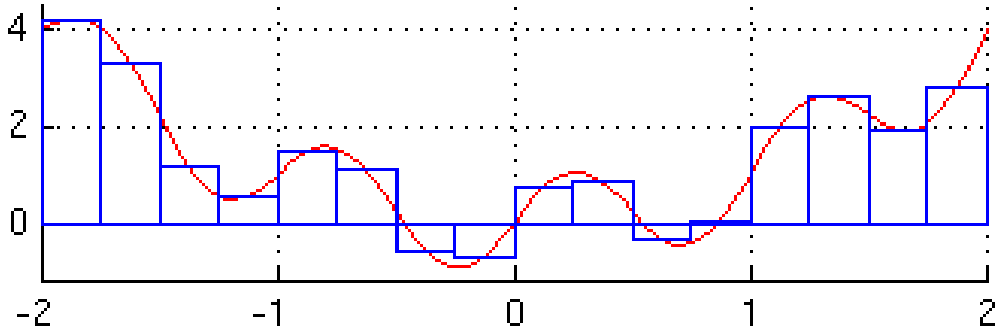
A continuación se muestra un algoritmo que permite obtener la derivada de manera que el número de valores obtenidos no difiera del de la función sobre la que se opera.

El método es una ligera modificación del algoritmo para la integral cumulativa, añadiendo el valor cero para el primer valor del resultado.

Si la función es F(x), su integral *cumulativa* es:

```
intF(1)=0 ;
intF(2:end)=cumsum(((F(1:end-1)+F(2:end))/2).*diff(x)) ;
```

Y la integral definida es el valor intF(end)



Ejercicio cinemática 1D:

Determinar, durante el intervalo de tiempo desde t=-1 s hasta t=22 s la posición $x(t)$ de una partícula cuya velocidad es $v_x(t)=-3.2*\cos(1.23*t)$ m/s, que parte de la posición $x(t=-1\ s)$=-0.15 m.

a) Representar gráficamente x(t) (desde t=-1 s hasta t=22 s).

b) Obtener la distancia recorrida en función del tiempo (desde t=-1 s hasta t=22 s).

```
%% velocidad
N=1000 ; % número de puntos
t1=-1 ; t2=22 ; % intervalo de tiempo
t=linspace(t1,t2,N) ;
vx=-3.2*cos(1.23*t) ;

%% posición x(t) (integral cumulativa de vx)
intx=0*vx ; intx(2:end)=cumsum(diff(t).*(vx(1:end-1)+vx(2:end))/2) ;
x=-0.15+intx ; % ATENCIÓN a la posición inicial

plot(t,x) ;
xlabel('tiempo (s)') ; ylabel('posición (m)') ;

%% distancia recorrida (ejercicio)
```

EJERCICIO: Considerar una partícula de masa m=100 gramos, que puede moverse sólo en la dimensión x, sobre la que actúa una fuerza (unidades Newton) Fx(t)=8.5*exp(-t/a)*sin(t-b), donde a=10 s, b=1.5 m. La posición inicial (t=0 s) de la partícula es x0=-1 m y su velocidad inicial es vx0=-10.1 m/s.
Determinar la posición de la partícula, en función del tiempo en el intervalo de 0 a 5 segundos, y representarla gráficamente.

```
% Parámetros y condiciones iniciales
m=0.1 ; a=10 ; b=1.5 ; x0=-1 ; vx0=-10.1 ; t0=0 ; t1=5 ;
% tiempo
n=1000 ; t=linspace(t0,t1,n) ; % n intervalos dt
% aceleración
ax=8.5*exp(-t/a).*sin(t-b)/m;

% integrar ax para obtener la velocidad vx(t).

% integrar vx para obtener la posición x(t).

% Representación gráfica
```

Ejercicio cinemática 2D

Considerar una partícula que se mueve en el plano *XY* según las componentes de la velocidad $v_x(t)=5*cos(2*t)$ m/s y $v_y=7*sin(1.23*t)+0.0003$ m/s, y que parte del punto (*x,y*)=(1.5,1.4) m en el instante *t*=2 s.

Considerando el intervalo de tiempo desde *t*=2 s hasta *t*=10 s:

a) Determinar las componentes de posición *x(t)* e *y(t)* y representarlas en el mismo gráfico (en función de *t*)
b) Representar gráficamente la trayectoria de la partícula (y(x)).
c) Determinar las componentes de la aceleración $a_x(t)$ y $a_y(t)$, y representarlas en el mismo gráfico.
d) Determinar la distancia al origen (*x,y*)=(0,0) m en función del tiempo, y representarla gráficamente.
e) Determinar la energía cinética de la partícula en función del tiempo si su masa es 0.34 kg, y representarla gráficamente.
f) Determinar la distancia recorrida por la partícula en función del tiempo, y representarla gráficamente.
g) Obtener el módulo de la aceleración tangencial en función del tiempo, y representarlo gráficamente.

EJERCICIO: Las coordenadas iniciales, *x*, de un conjunto de 20 partículas vienen dadas por la expresión, en metros, (para la partícula *N* desde 0 hasta 19).

$$x(N) = 2 \times \sum_{n=0}^{N} \frac{2^n (n!)^2}{(2n+1)!}$$

Representar gráficamente *x*(*N*) y comprobar que la posición tiende asintóticamente hacia 3.14 metros según aumenta *N*.

```
N=[0:1:19] ; % índice de partícula
% ATENCIÓN a los operadores elemento a elemento
x_serie=2*2.^N.*(factorial(N)).^2./factorial(2*N+1) ;
x=cumsum(x_serie) ;
plot(N,x,'og') ;
% title, xlabel, ylabel, etc.
```

Ejercicio: representar en la misma gráfica *x*(*N*) e *y*(*N*), si *y*(*N*)=2**x*(*N*), en distintos colores y añadir la leyenda.

Exportar datos del workspace de variables a un fichero .mat

Todas las variables se encuentran en el *workspace*, que es visible en su ventana.

Se guarda todo el *workspace* en un fichero con extensión *.mat*, en un formato sólo legible por MATLAB.

```
>> save('nombre_fichero.mat')
```

Guardar sólo algunas variables

```
>> save('nombre_fichero.mat','var1','var2')
```

Añadir variables a un fichero existente

```
>> save('nombre_fichero.mat','var3','-append')
```

Importar datos de un fichero .mat al workspace (añadir o sustituir)

```
>> load('nombre_fichero.mat')
>> load('nombre_fichero.mat','var1','var2')
```

Importar/exportar datos en fichero de texto ASCII

```
>> var=importdata('nombre_fichero.txt')
```

Función **`save`** con opción –ascii para guardar datos en formato texto.

COMANDOS, FUNCIONES, OPERADORES

- Operadores aritméticos: `+ - * / ^ .* ./ .^`
- Funciones: `sin cos tan asin acos atan sqrt log log10 exp factorial sign round fix floor ceil max min rem mod`
- Operaciones con vectores: `[] () linspace length diff sum cumsum prod cumprod dot cross norm`
- Operador *dos puntos* para generar vectores.
- Variables predefinidas `pi ans end inf nan`
- Variables de texto: entre comillas simples
- Figuras: `figure close plot subplot xlabel ylabel title legend xlim ylim grid`
- Otros: `; , … % %% clear help who whos format clc`
- Importar y exportar datos: `save load importdata`

DESTREZAS

- Generar vectores utilizando el operador *dos puntos* `:` `linspace logspace`.
- Acceder a elementos individuales con `() end`.
- Aplicar funciones a vectores. Producto escalar y vectorial de vectores.
- Operaciones algebraicas con vectores. Operadores *elemento a elemento* (con punto).
- Escribir y ejecutar ficheros *script* (*.m*) con secciones (y documentados!)
- Calcular la derivada numérica, ejercicios de cinemática.
- Calcular la integral numérica (definita y cumulativa), ejercicios de cinemática.
- Pintar funciones y trayectorias (`plot subplot` etc.), etiquetar ejes, formato visual de datos, etc.
- Importar y exportar variables del workspace a ficheros *.mat*.

ATENCIÓN a los ejemplos y ejercicios propuestos en el aula

PREGUNTAS, DUDAS, COMENTARIOS

% Las variables asignadas con = NO se ejecutan automáticamente!!!! (como en Excel y similares)
% Esto es, el operador = da lugar a la asignación de un valor, no a una relación entre variables

```
>> a=3 , b=2*a , a=4   % b sigue teniendo el valor 6 (NO 8). Haría falta b=2*a otra vez.
```

% Mostrar valor de variables con función disp

```
a=[4 6 9.7] ;
disp('a') ; disp(a) ;
disp(['a= ',num2str(a)]) ;
```

Número de elementos de un vector

```
>> a=[4;5;6] ; length(a)
```

Conversión de vector fila en columna y viceversa: **`transpose`** , o apóstrofe ' después del vector:

```
>> a=transpose([4 7]) , b=[4 7]'
```

Sub- y Súper-índices en texto con caracteres _ ^ _{ij} ^{-10}

Menú->Home->**Layout** para establecer ventanas del desktop

SYMBOLIC MATH (ToolBox con MuPAD) NO FORMA PARTE DEL CURSO pero puede resultar útil para otras asignaturas

TAYLOR EXPANSION

```
% taylor expansion of a one variable function
syms x % create a symbolic variable named x
taylor(exp(x)) % see help taylor
taylor(exp(x),x,'ExpansionPoint', 1)
taylor(exp(x), 'Order', 10)

% taylor of a two variable function
syms x y ; taylor(exp(x)*sin(y),y)  % with respect to y

% integration and differentiation
>> syms x , int(sin(x))
>> syms x , diff(sin(x))
```