Física
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
UNIVERSIDAD DE CHILE
FI1002 - Sistemas Newtonianos

Introducción a MATLAB

Auxiliar: Jonathan Monsalve Reyes

Operaciones Básicas

- Las operaciones aritméticas básicas son: + - * / ^ que pueden combinarse con paréntesis. El orden de prioridad en la evaluación de una expresión aritmética es:

1. Cantidades entre paréntesis
2. ^
3. * / de izquierda a derecha
4. + - de izquierda a derecha

Reconoce los siguientes tipos de variables:

reales	3.1415, 1.42e+8
complejas	1.23 + 5.98i, 1 + 1i
caracter	`'entre_comillas simples'`
lógicas	true (1), false (0)

Operaciones Básicas

- MATLAB trabaja internamente con aritmética IEEE (i.e.15 cifras significativas). La forma de imprimir los resultados en pantalla se controla con la sentencia **format.** De igual forma se puede modificar el formato desde la opción *Preferences*, menú *File*.

`format short`	3.1416
`format long`	3.14159265358979
`format short g`	La mejor de las dos anteriores con 5 dígitos
`format short e`	3.1416e+00
`format long e`	3.14159265358979e+00
`format long g`	La mejor de las dos anteriores con 15 dígitos
`format hex`	400921fb54442d18
`format bank`	3.14
`format rat`	355/113

Operaciones Básicas

- Los nombres de variables pueden contener cualquier combinación de letras, números y el símbolo _ , aunque siempre comenzando por una letra. MATLAB tiene variables predefinidas como:

`ans`	último cálculo no asignado
`pi`	$\pi = 3.141\ldots$
`i, j`	$\sqrt{-1}$ unidad imaginaria
`Inf, NaN`	infinito y NaN (aritmética IEEE)
`eps`	distancia del 1 hasta el siguiente número en coma flotante $2^{-52} \simeq 2.22 \times 10^{-16}$ (doble de la unidad de redondeo)
`realmax`	$\simeq 1.80 \times 10^{308}$
`realmin`	$\simeq 2.23 \times 10^{-308}$

Notas

- Si queremos suprimir la salida de las variables por pantalla, hay que añadir un “;” al final de cada sentencia. Esto es útil cuando se ejecuta un fichero de órdenes en MATLAB. Pueden escribirse varias sentencias en una línea separadas por “;” o “,”.
- La sentencia **clc** borra la ventana de comandos.
- Si no se asigna un resultado a una variable, por defecto lo asocia a la variable **ans**.
- Los comentarios en MATLAB deben ir precedidos por el símbolo %.
- La orden **clear all** borra el contenido de todas las variables y funciones.

Utilidades

- Emplear **help comando** o **helpwin comando** permite obtener información sobre **comando.**
- Para abortar una tarea en MATLAB utilizar CTRL + c.
- Para medir el tiempo de cpu consumido por un bloque de sentencias pueden usarse:

```
>> tic; bloque de sentencias; toc
>> ti=cputime; bloque de sentencias; cputime-ti
```

- Entrada de datos por teclado: Para este fin está la función **input**, con sintaxis:

```
>> variable= input('texto ');
texto 2.345
>> variable=input('texto ','s');
```

siendo la primera para valores numéricos y la segunda para variables tipo carácter.

Utilidades

- Salida de datos formateados: La función adecuada es:

  fprintf('format',vars)

  donde **vars** es una lista de variables separadas por comas y algunos de los elementos que pueden escribirse en **format** vienen dados en la tabla siguiente:

%P.Qe	notación exponencial
%P.Qf	notación decimal
%P.Qg	la más corta de las anteriores
%Pd, %Pi	imprime un entero
%s	imprime una variable caracter
\n	salto de línea

Utilidades

- Con P y Q números enteros no negativos. P indica la longitud mínima de caracteres que se usará para escribir el número y Q el número de dígitos de la parte decimal.

```
>> x=1007.46; y=pi; j=78;
>> fprintf( 'x= %8.2e y=%12.8f j=%i\n', x,y,j)
    x= 1.01e+03 y= 3.14159265 j=78
```

- La sentencia **disp(X)** es una forma simple de imprimir la matriz X sin su nombre o un texto si X es una variable caracter.

```
X =
   1.0000   0.5000   0.3333   0.2500
   0.5000   0.3333   0.2500   0.2000
   0.3333   0.2500   0.2000   0.1667
   0.2500   0.2000   0.1667   0.1429
>> disp(X); disp('imprime esto');
   1.0000   0.5000   0.3333   0.2500
   0.5000   0.3333   0.2500   0.2000
   0.3333   0.2500   0.2000   0.1667
   0.2500   0.2000   0.1667   0.1429

imprime esto
```

Funciones Predefinidas

- **help elfun**

```
Trigonometric.
 sin         - Sine.
 sinh        - Hyperbolic sine.
 asin        - Inverse sine.
 asinh       - Inverse hyperbolic sine.
 cos         - Cosine.
 cosh        - Hyperbolic cosine.
 acos        - Inverse cosine.
 acosh       - Inverse hyperbolic cosine.
 tan         - Tangent.
 tanh        - Hyperbolic tangent.
 atan        - Inverse tangent.
 atan2       - Four quadrant inverse tangent.
 atanh       - Inverse hyperbolic tangent.
```

Funciones Predefinidas

```
  sec         - Secant.
  sech        - Hyperbolic secant.
  asec        - Inverse secant.
  asech       - Inverse hyperbolic secant.
  csc         - Cosecant.
  csch        - Hyperbolic cosecant.
  acsc        - Inverse cosecant.
  acsch       - Inverse hyperbolic cosecant.
  cot         - Cotangent.
  coth        - Hyperbolic cotangent.
  acot        - Inverse cotangent.
  acoth       - Inverse hyperbolic cotangent.

Exponential.
  exp         - Exponential.
  log         - Natural logarithm.
  log10       - Common (base 10) logarithm.
  log2        - Base 2 logarithm and dissect floating
                point number.
  pow2        - Base 2 power and scale floating
                point number.
  sqrt        - Square root.
  nextpow2    - Next higher power of 2.
```

Funciones Predefinidas

```
Complex.
  abs         - Absolute value.
  angle       - Phase angle.
  complex     - Construct complex data from real
                and imaginary parts.
  conj        - Complex conjugate.
  imag        - Complex imaginary part.
  real        - Complex real part.
  unwrap      - Unwrap phase angle.
  isreal      - True for real array.
  cplxpair    - Sort numbers into complex conjugate pairs.

Rounding and remainder.
  fix         - Round towards zero.
  floor       - Round towards minus infinity.
  ceil        - Round towards plus infinity.
  round       - Round towards nearest integer.
  mod         - Modulus (signed remainder after division).
  rem         - Remainder after division.
  sign        - Signum.
```

Vectores y Matrices

- En MATLAB tienen un lugar privilegiado los vectores y matrices. De hecho, podríamos considerar que es el único tipo de variable que existe (un escalar se considera como una matriz 1 × 1).

```
>> vector_fila = [1,2,3]
vector_fila =
  1  2  3
>> vector_columna = [1;2;3]
vector_columna =
  1
  2
  3
>> matriz = [11,12,13; 21,22,23; 31,32,33]
matriz =
  11  12  13
  21  22  23
  31  32  33
```

Vectores y Matrices

- Los elementos de una matriz se introducen por filas, separándolas con ; y en cada fila se separan por comas o espacios en blanco. La notación **a:b:c** sirve para construir vectores fila comenzando en **a**, incrementándose por el valor **b** hasta **c**. El comando **whos** lista las variables empleadas, el tamaño, espacio que ocupan y la clase.

```
>> x=1:1:10, y=1:10
x =
  1  2  3  4  5  6  7  8  9  10
y =
  1  2  3  4  5  6  7  8  9  10
>> z=0.1:.05:.3, x1=-1:-2, p=1;
z =
  0.1000  0.1500  0.2000  0.2500  0.3000
x1 =
  Empty matrix:  1-by-0
>> whos
  Name   Size   Bytes   Class

  p      1x1        8   double array
  x      1x10      80   double array
  x1     1x0        0   double array
  y      1x10      80   double array
  z      1x5       40   double array
Grand total is 25 elements using 200 bytes
```

Vectores y Matrices

- Para convertir un vector fila en un vector columna (o recíprocamente) se utiliza la transposición, que viene denotada por ‘ .

```
>> matriz'
ans =
  11 21 31
  12 22 32
  13 23 33
>> matriz_compleja=[ 1+1i, 1+2i; 2+1i, 2+2i]
matriz_compleja =
  1 + 1i 1 + 2i
  2 + 1i 2 + 2i
>> matriz_compleja' % compleja transpuesta conj.
ans =
  1 - 1i 2 - 1i
  1 - 2i 2 - 2i
>> matriz_compleja.'  % compleja transpuesta
ans =
  1 + 1i 2 + 1i
  1 + 2i 2 + 2i
```

Vectores y Matrices

- Podemos efectuar las siguientes operaciones entre matrices, siempre que estén definidas:

`c=a ± b`	suma (resta) ($c_{ij} = a_{ij} \pm b_{ij}$) si `a` es escalar, ($c_{ij} = a \pm b_{ij}$)
`c=a*b`	producto matricial
`c=a.*b`	producto de matrices elemento a elemento ($c_{ij} = a_{ij} * b_{ij}$)
`c=a/b`	equivalente a `c=(inverse(b')*a')'`
`c=a./b`	cociente elemento a elemento ($c_{ij} = a_{ij}/b_{ij}$) si `a` es escalar, ($c_{ij} = a/b_{ij}$)
`c=a\b`	equivalente a `c=inverse(a)*b`
`c=a^k`	potencia k–ésima de la matriz `a`
`c=a.^b`	potenciación elemento a elemento ($c_{ij} = a_{ij}^{b_{ij}}$)
`c=a'`	matriz compleja conjugada transpuesta (`ctranspose(a)`)
`c=a.'`	matriz transpuesta (`transpose(a)`)

Vectores y Matrices

- Hay varias funciones que permiten manejar y definir matrices de forma cómoda, como:

`ones(nf,nc)`	matriz de unos de dimensión `nf`×`nc`
`zeros(nf,nc)`	matriz de ceros de dimensión `nf`×`nc`
`eye(nf,nc)`	matriz identidad de dimensión `nf`×`nc`
`[nf,nc]=size(A)`	número de filas y de columnas de la matriz `A`
`diag(v)`	matriz diagonal con vector `v` en la diagonal
`hilb(n)`	matriz de Hilbert de dimensión `n`×`n`
`norm(A,p)`	norma `p` de `A`. `p=1,2,Inf,'fro', ...`
`rand(nf,nc)`	matriz de números aleatorios uniformemente distribuidos en el intervalo (0,1) de dimensión `nf`×`nc`
`randn(nf,nc)`	matriz de números normalmente distribuidos con media nula y varianza uno de dimensión `nf`×`nc`

Vectores y Matrices

`sum(A)`	si `A` es una matriz `nf` $\times$ `nc`, el resultado es un vector fila `s` con $\mathtt{s}_i = \sum_{j=1}^{\mathtt{nf}} A(j,i)$, $i = 1, \ldots, \mathtt{nc}$. Si `A` es un vector, es la suma de las componentes de `A`.
`prod(A)`	análogo a `sum` pero con productos.
`max(A)`	si `A` es una matriz `nf` $\times$ `nc`, el resultado es un vector fila `s` con $\mathtt{s}_i = \max_{1 \le j \le \mathtt{nf}} A(j,i)$, $i = 1, \ldots, \mathtt{nc}$. Si `A` es un vector, es el máximo de las componentes de `A`.
`min(A)`	análogo a `max` pero con mínimo. Si `A` es compleja ver `help max`
`linspace(a,b,n)`	genera una red de `n` puntos igualmente espaciados entre `a` y `b`.
`logspace(a,b,n)`	genera una red de `n` puntos logarítmicamente espaciados entre $10^{\mathtt{a}}$ y $10^{\mathtt{b}}$.
`repmat(A,nf,nc)`	genera una matriz por bloques `nf`$\times$`nc` con el valor de `A` en cada bloque.

Iteraciones

- Sentencia for:

  for i=i0:ipaso:ifinal

  % expresiones lógicas

  end

```
>> for i=[1,3,5,8:12]; i, end % imprime i
>> for i=1:10; i, end % imprime i
```

- Sentencia if:

```
if condición
  bloque de sentencias if
elseif condición1
  bloque de sentencias elseif condición1
elseif condición2
  bloque de sentencias elseif condición2
else
  bloque de sentencias else
end
```

Iteraciones

- Sentencia **while**:

```
while condición
  bloque de sentencias
end
```

que ejecuta el bloque de sentencias mientras la condición sea cierta. **break** termina la ejecución de bucles **for** o **while**. En bucles encajados, termina la ejecución sólo del bucle más interno.

Script y funciones: m-file

- Un **script m-file** permite almacenar comandos MATLAB y poder ejecutarlos posteriormente, ... Prácticamente muchos de los ejemplos anteriores se podrían haber escrito en este formato.
- Una **función m-file** permite escribir funciones estilo MATLAB de manera que acepten argumentos de entrada y de salida, siendo su forma general:

```
function [variables_out] = nombre_funcion(variables_in)
Definición de la función
```

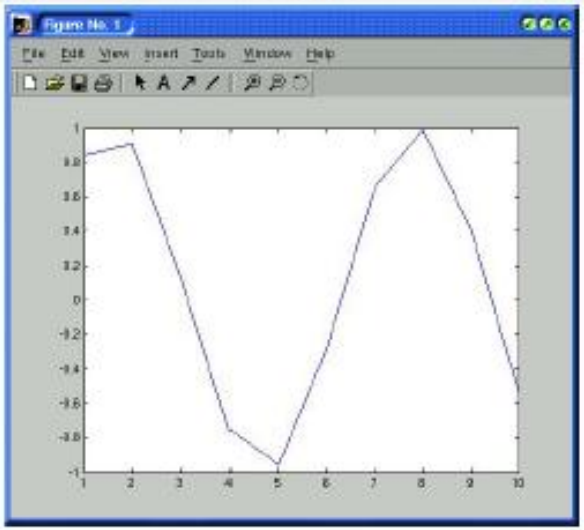
- Puede emplearse **return** para salir de una función.

```
function [volumen, area] = esfera(r)
% esta funcion calcula el volumen
% y el area de una esfera de radio r
volumen = 4/3*pi*r^3;
area = 4*pi*r^2;
```

Gráficos

- 2D: El dibujo más sencillo se realiza con **plot(x,y)**, siendo x e y vectores, que dibuja la poligonal que une los puntos (*xi*, *yi*), *i* = 1, . . . , *n*.

```
>> x=1:10; y=sin(x); plot(x,y)
```



Gráficos

- También existen las siguientes instrucciones útiles:

```
>> clf % borra el contenido de la ventana gráfica
>> hold on
>> plot (...)  % varios dibujos en la misma ventana
>> plot (...)
>> hold off
>> grid on % Superpone una rejilla
>> grid off % Borra la rejilla
```

- axis: Control de ejes:

```
axis([xmin xmax ymin ymax])
```

Gráficos

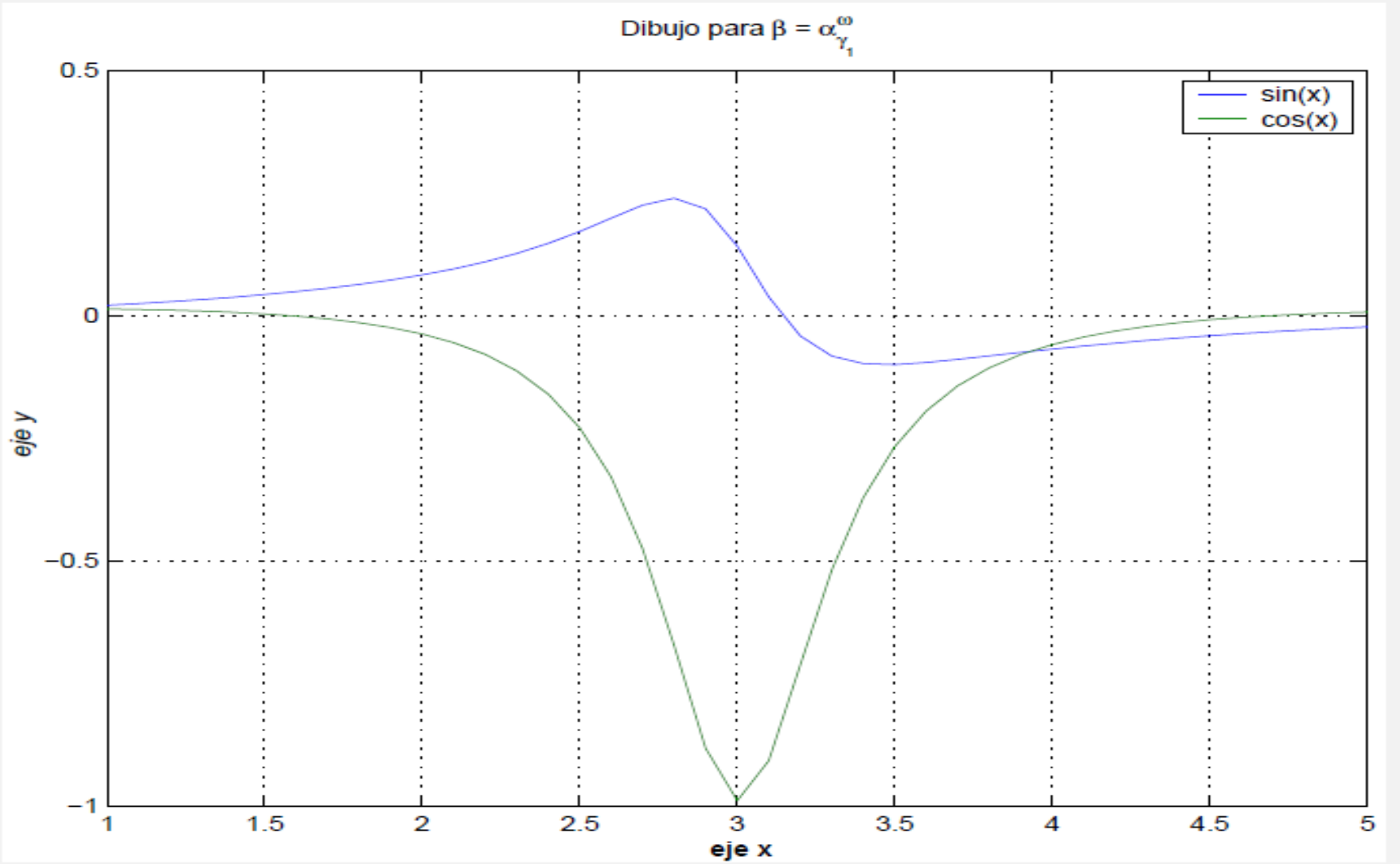
- **text(x,y,'texto')**: Añade texto al gráfico en el punto de coordenadas **(x,y)**.
- **title('titulo')**: Añade **título** en la parte superior del dibujo.
- **xlabel('etiqueta x')**: Añade **etiqueta x** en el eje de abcisas.
- **ylabel('etiqueta y')**: Análogo a **xlabel**, pero para el eje de ordenadas.
- **legend('text1','text2',...,pos)**: Añade al gráfico las leyendas **text1**, **text2**, . . . como etiquetas. La colocación depende del valor de pos:

0	Automático (Defecto).
1	Esquina superior derecha
2	Esquina superior izquierda
3	Esquina inferior derecha
4	Esquina inferior izquierda
-1	Derecha del gráfico

Gráficos

```
>> x=1:0.1:5;
>> y = 1./(1 + 10*(x-3).^2).*sin(x);
>> z = 1./(1 + 10*(x-3).^2).*cos(x);
>> clf;
>> plot(x,y,x,z)
>> title('Dibujo para \beta = \alpha_{\gamma_1}^\omega')
>> xlabel('\bf eje x'); ylabel('\it eje y')
>> legend('sin(x)','cos(x)');
>> grid on
```

Gráficos

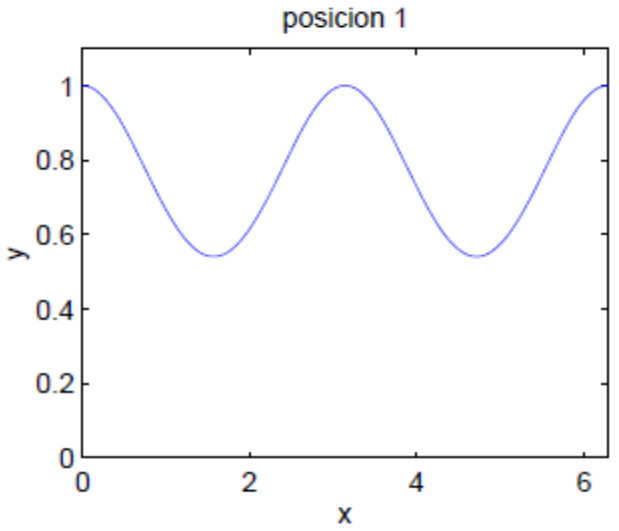

Dibujo para β = $\alpha_{\gamma_1}^{\omega}$
sin(x)
cos(x)
0.5
0
−0.5
−1
1
1.5
2
2.5
3
3.5
4
4.5
5
eje y
eje x

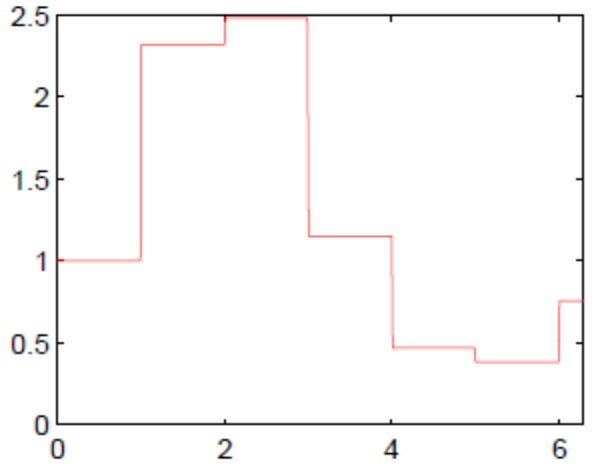
Gráficos

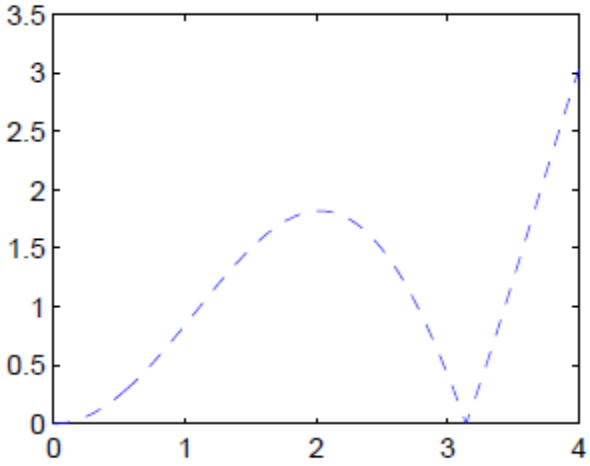
- La ventana gráfica de MATLAB puede albergar varios gráficos independientes. Esto se hace con la orden **subplot(mnk)** o **subplot(m,n,k)**. Esta divide la ventana gráfica en una matriz **m**×**n** de subventanas, realizando el dibujo en la ventana **k**, contando por filas.
- Para dibujar expresiones matemáticas, puede resultar preferible emplear la orden **fplot ('funcion', dominio)**, que genera de manera automática el dibujo.

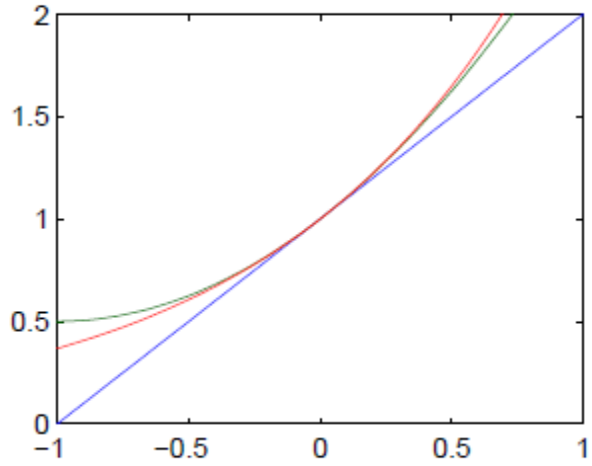
```
>> subplot(221); fplot('cos(sin(x))',[0 2*pi 0 1.1])
>> xlabel('x'); ylabel('y'); title('posicion 1')
>> subplot(222);
>> fplot('exp(sin(floor(x)))',[0 2*pi],'r')
>> subplot(223);
>> fplot('abs(sin(x)*x)',[0 4],'b--')
>> subplot(224);
>> fplot('[1+x,1+x+x^2/2,exp(x)]',[-1 1 0 2])
```

Gráficos


posicion 1
y
x
0
0.2
0.4
0.6
0.8
1
0
2
4
6


0
0.5
1
1.5
2
2.5
0
2
4
6


0
0.5
1
1.5
2
2.5
3
3.5
0
1
2
3
4


0
0.5
1
1.5
2
−1
−0.5
0
0.5
1

Gráficos

- La ventana gr´afica de MATLAB puede albergar varios gráficos independientes. Esto se hace con la orden **subplot(mnk)** o **subplot(m,n,k).** Esta divide la ventana gráfica en una matriz **m**×**n** de subventanas, realizando el dibujo en la ventana **k**, contando por filas.
- Para dibujar expresiones matemáticas, puede resultar preferible emplear la orden **fplot ('funcion', dominio)**, que genera de manera automática el dibujo.

```
>> subplot(221); fplot('cos(sin(x))',[0 2*pi 0 1.1])
>> xlabel('x'); ylabel('y'); title('posicion 1')
>> subplot(222);
>> fplot('exp(sin(floor(x)))',[0 2*pi],'r')
>> subplot(223);
>> fplot('abs(sin(x)*x)',[0 4],'b--')
>> subplot(224);
>> fplot('[1+x,1+x+x^2/2,exp(x)]',[-1 1 0 2])
```

¿Consultas?