MÉTODOS NUMÉRICOS
GUÍA DE LABORATORIO NRO. 5
GRÁFICOS E INTERPOLACIÓN EN MATLAB

OBJETIVOS:

Familiarizarse con los comandos y entorno para realizar gráficos en MATLAB.
Utilizar comandos y funciones para interpolación.
Resolver problemas mediante el uso del MATLAB.

MEDIOS Y MATERIALES EDUCATIVOS:

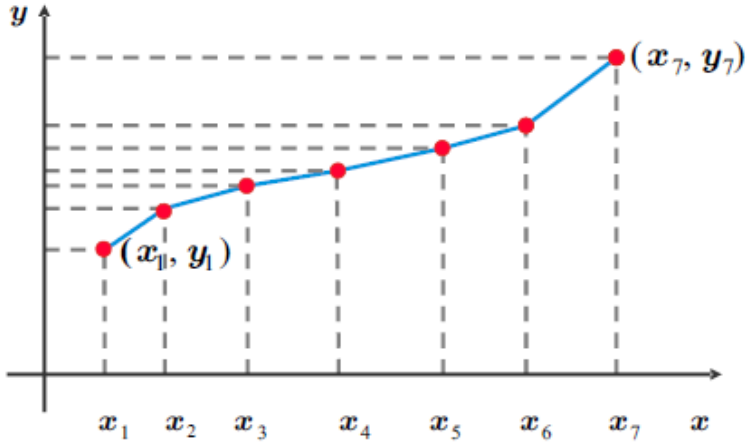
Guía de laboratorio, computadora, software de Matlab, tutoriales y manuales de Matlab, apuntes, Internet y flash memory.

INFORME:

Realizar un informe del laboratorio realizado, puede ser individual o de un máximo de dos estudiantes.

TAREA 1. GRÁFICOS EN MATLAB

La representación gráfica de una curva en un ordenador es una línea poligonal construida uniendo mediante segmentos rectos un conjunto discreto y ordenado de puntos: {(x1, y1), (x2, y2), . . . , (xn, yn)}.



La línea así obtenida tendrá mayor apariencia de «suave» cuanto más

puntos se utilicen para construirla, ya que los segmentos serán imperceptibles.
Para dibujar una curva plana en MATLAB se usa el comando

plot(x,y)

siendo x e y dos vectores de las mismas dimensiones conteniendo, respectivamente, las abscisas y las ordenadas de los puntos de la gráfica.

Ejemplo.

Dibujar la curva $y = \frac{x^2 + 2}{x + 5}$ para x ∈ [−2, 3]

En Matlab:

```
>> f = @(x) (x.^2+2)./(x+5);
>> x = linspace(-2,3);
>> plot(x,f(x))
```

Se pueden dibujar dos o más curvas de una sóla vez, proporcionando al comando plot varios pares de vectores abscisas-ordenadas, como en el ejemplo siguiente.

Ejemplo.

Dibujar las curvas $y = 2\,\text{sen}^3(x)\cos^2(x)$ e $y = e^x - 2x - 3$ para x ∈ [−1.5, 1.5].

En Matlab:

```
>> f1 = @(x) 2 * sin(x).^3 .* cos(x).^2;
>> f2 = @(x) exp(x) - 2*x -3;
>> x = linspace(-1.5,1.5);
>> plot(x,f1(x),x,f2(x))
```

A cada par de vectores abscisas-ordenadas en el comando plot se puede añadir un argumento opcional de tipo cadena de caracteres, que modifica el aspecto con que se dibuja la curva. Para más información, se puede digitar help plot.
Ejemplo.

Las siguientes órdenes dibujan la curva $y = sen^3(x)$ en color rojo (r, de red) y con marcadores $*$, en lugar de una línea continua.

```
>>  x  =  linspace(0,pi,30);
>>  plot(x,sin(x).^3,'r*')
```

La orden siguiente dibuja la curva $y = sen^3 x$ en color negro (k, de black), con marcadores $*$ y con línea punteada, y la curva $y = cos^2 x$ en color azul (b, de blue) y con marcadores +.

```
>>  plot(x,sin(x).^3,'k*:',x,cos(x).^2,'b+')
```

Además, mediante argumentos opcionales, es posible modificar muchas otras propiedades de las curvas. Esto se hace siempre mediante un par de argumentos en la orden de dibujo que indican

'Nombre de la Propiedad', Valor de la propiedad

Esta propiedad afectará a todas las curvas que se dibujen con la misma orden.

Ejemplo.

Para establecer un grosor determinado de las líneas se usa la propiedad LineWidth (atención a las mayúsculas):

```
>>  plot(x,sin(x).^3,'k*:',x, cos(x).^2,'b+','LineWidth',1.1)
```

Se pueden añadir elementos a la gráfica, para ayudar a su comprensión. Para añadir una leyenda que identifique cada curva se usa el comando siguiente, que asigna las leyendas a las curvas en el orden en que han sido dibujadas.

legend('Leyenda1', 'Leyenda2')

Para añadir etiquetas a los ejes que aclaren el significado de las variables se usan los comandos:

xlabel('Etiqueta del eje OX')
ylabel('Etiqueta del eje OY')

Se puede añadir una cuadrícula mediante la orden:

grid on

También es muy útil la orden, siguiente, que define las coordenadas mínimas y máximas del rectángulo del plano OXY que se visualiza en la gráfica:

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

Cada nueva orden de dibujo borra el contenido previo de la ventana gráfica, si existe. Para evitar esto existen la órdenes:

hold on
hold off

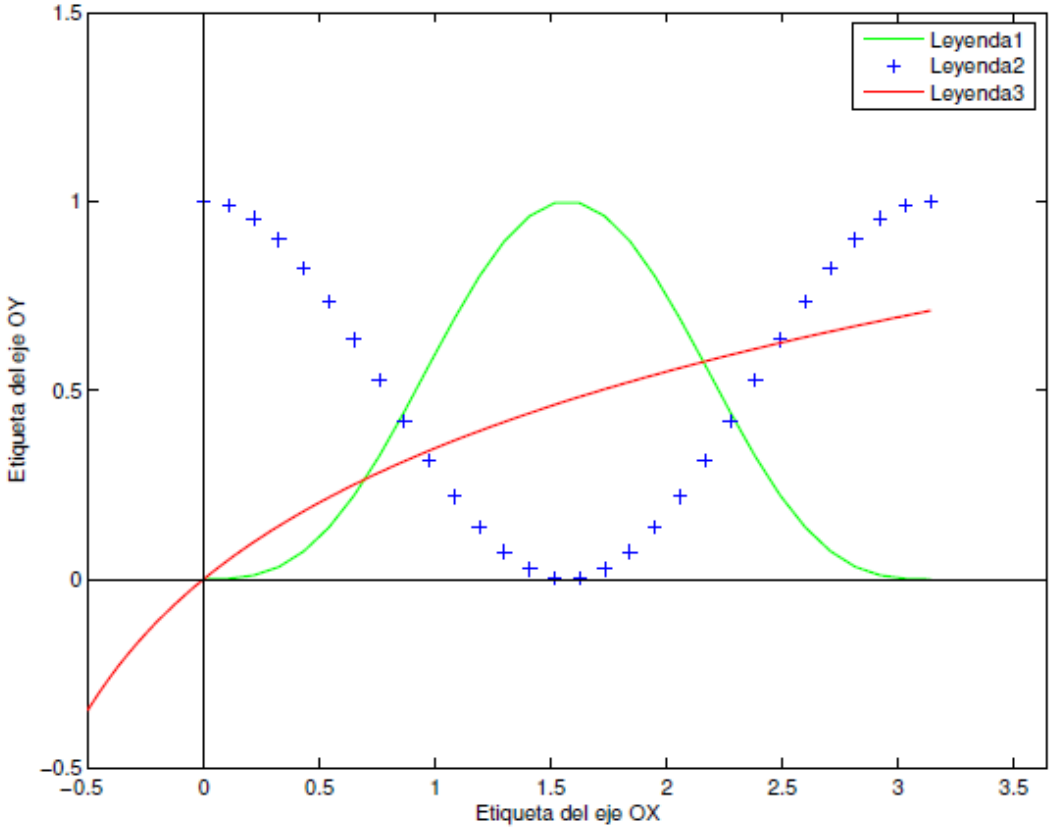
La orden hold on permanece activa hasta que se cierre la ventana gráfica o bien se dé la orden hold off

Ejemplos.

Las siguientes órdenes darán como resultado la gráfica de la figura siguiente:

```
x = linspace(0,pi,30);
axis([-0.5,pi+0.5,-0.5,1.5])
hold on
plot(x,sin(x).^3,'g', x, cos(x).^2,'b+', 'LineWidth', 1.1)
x = linspace(-0.95, pi);
plot(x, log(x+1)/2, 'r', 'LineWidth', 1.1)
plot([-5,5], [0, 0], 'k', 'LineWidth', 1)
plot([0, 0], [-5,5], 'k', 'LineWidth', 1)
legend('Leyenda1', 'Leyenda2', 'Leyenda3')
xlabel('Etiqueta del eje OX')
ylabel('Etiqueta del eje OY')
hold off
shg
```

Figura: Varias curvas con leyenda y etiquetas.

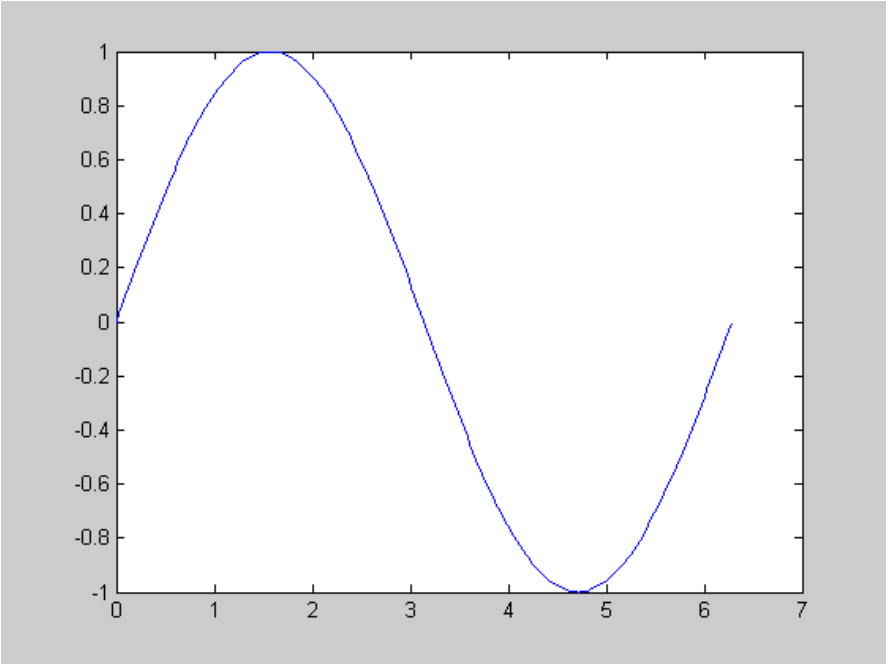


Gráficos básicos.

Más ejemplos de la capacidad de MATLAB de realizar representaciones gráficas.

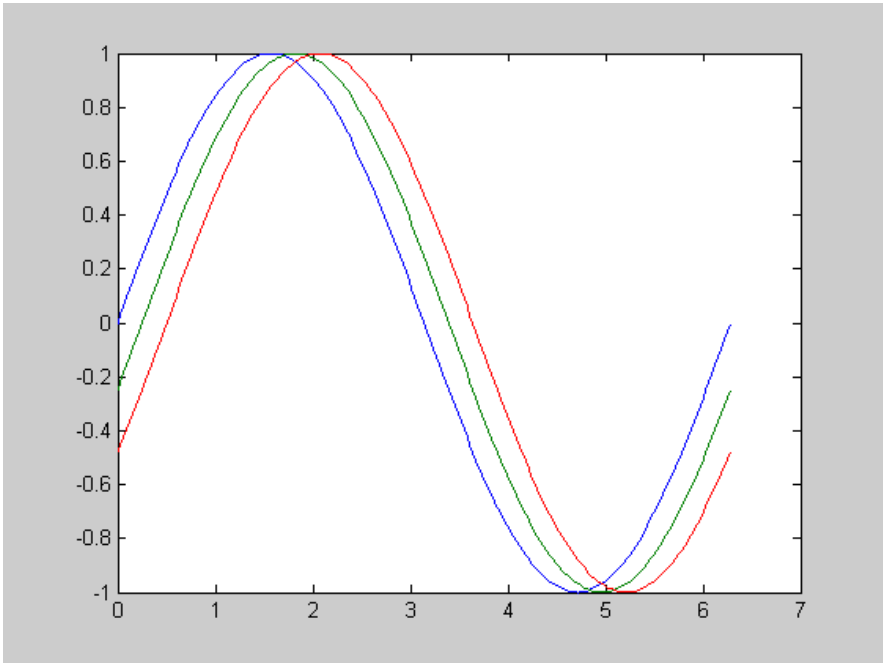
```
>> t=0:pi/100:2*pi;      % Define el eje de tiempos como un vector de
                         % 200 puntos entre 0 y 6.28.
>> y=sin(t);             % Calcula el seno.
>> plot(t,y);            % Representación del seno frente al tiempo.
```

La siguiente figura muestra el resultado de ejecutar la última instrucción.



Para representar varias señales a la vez, puede realizar el siguiente conjunto de instrucciones:

```
>> y2=sin(t-0.25);
>> y3=sin(t-0.5);
>> plot(t,y,t,y2,t,y3);
```

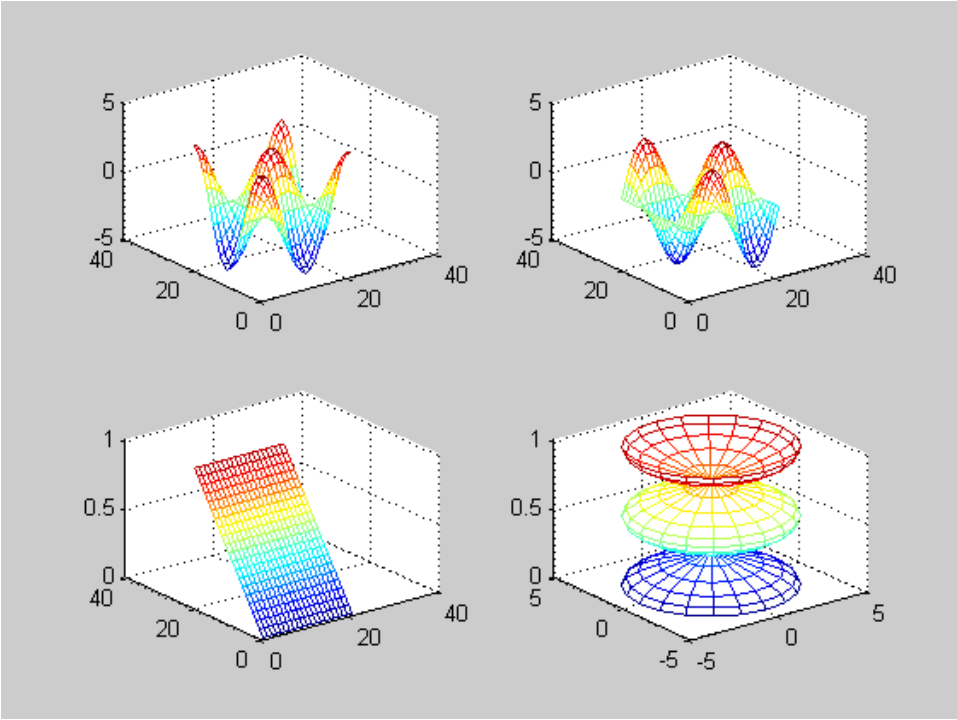


Consultar la ayuda para conocer los distintos colores, tipos de línea, etc...

que se puede emplear.
El comando figure sirve para crear una nueva figura o para invocar figuras ya existentes.
El comando hold on permite realizar un nuevo gráfico sobre otro ya existente sin que se abra una nueva figura. Si es preciso, se reescalan los ejes. El comando hold off desactiva la opción anterior, evitando que un nuevo gráfico se superponga a uno ya existente.

Se verá como subdividir la pantalla (en un ejemplo que emplea gráficos en tres dimensiones):

```
>> t=0:pi/10:2*pi;              % Definición del eje de tiempos
>> [X,Y,Z]=cylinder(4*cos(t));
>> subplot(2,2,1), mesh(X); % Dibujo en el primer cuadrante
>> subplot(2,2,2), mesh(Y);
>> subplot(2,2,3), mesh(Z);
>> subplot(2,2,4), mesh(X,Y,Z);
```



Se verá ahora como mejorar la apariencia de una gráfica:

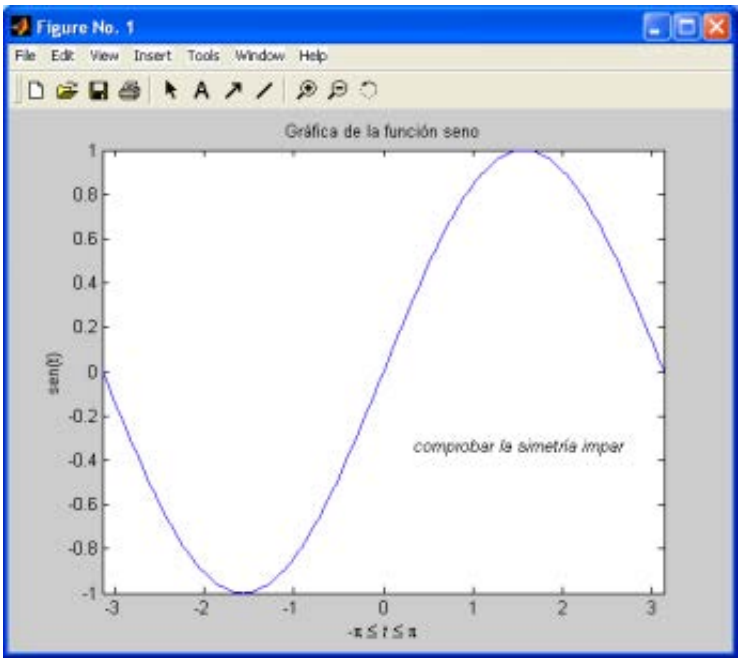
```
>> clf;                    % Borra la figura anterior
>> t=-pi:pi/100:pi;        % Eje temporal
```

```
>> y=sin(t);                % Señal a representar (sinusoide)
>> plot(t,y);               % Dibujo
>> axis([-pi pi -1 1]);     % Ejes
>> xlabel('-\pi \leq {\itt} \leq \pi');    % Etiquetado del eje horizontal
>> ylabel('sen(t)');                       % Etiquetado del eje vertical
>> title('Gráfica de la función seno');    % Nombre del gráfico
>> text(1/3,-1/3,'\it{comprobar la simetría impar}'); % Comentario
```

Observe el aspecto de la figura creada:

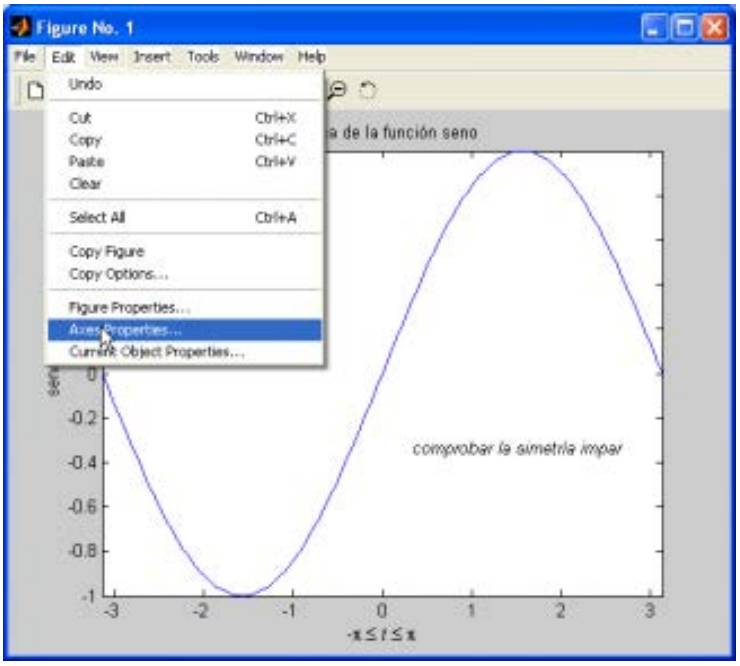
MATLAB encuentra los valores máximos de los valores a representar y escala los ejes de acuerdo a esos valores. El comando axis([xmin xmax ymin ymax]) permite especificar al usuario los valores de los ejes. Este comando axis, también admite una serie de parámetros que se recomienda consultar en la ayuda, como axis square, equal, on, offf,...

El comando grid on activa una rejilla en el dibujo, mientras que el comando grid off desactiva esa opción.



Edición de dibujos.

Cuando aparece una figura como la anterior, se puede emplear las opciones que aparecen en los menús desplegables de la parte superior y las herramientas de la barra para editar el dibujo, añadir texto, flechas,.... Se puede emplear el editor de propiedades que se encuentra en el menú EDIT de una figura para cambiar las propiedades de los distintos elementos de una figura.



TAREA 2. INTERPOLACIÓN

a) Analizar el siguiente código y explicar cómo funciona, y cómo se aplica la interpolación:

```
xdat = [1 5 10 30 50];
ydat = log(xdat);
plot(xdat,ydat, 'o')
hold on
pause
p = polyfit(xdat,ydat,2) % grado 2
xvet=1:0.1:50;
plot(xvet,polyval(p,xvet))
hold on
pause
p = polyfit(xdat,ydat,4) % grado 4
plot(xvet,polyval(p,xvet))
```

b) Consideremos estos precios del gas natural:

Año	x	2007	2008	2009	2010	2011	2012
Precio	y	133,5	132,2	138,7	141,5	137,6	144,2

Si se construye un pequeño script de MATLAB para analizar las prestaciones de la interpolación polinómica con estos datos resultaría una figura. A su lado se ve la sesión de ejecución del mismo y el gráfico a que da lugar. Algo no parece ir bien del todo, explicar.

c) Crear el script para la siguiente función:

```
function [l,L]=lagrang_int(x,y)
% Calcula el polinomio de interpolación de Lagrange
% x e y vectores de datos, l coef. polinomio resultante
% L polinomio base
N=length(x)-1; l=0;
for m=1:N+1
P=1;
for k=1:N+1
if k~=m, P=conv(P,poly(x(k)))/(x(m)-x(k)); end
end
L(m,:)=P; % Polinomios de Lagrange base
l=l+y(m)*P; % Coef. polinomio resultante
end
```

```
end
```

Analizar como interpola por el método de Lagrange, y comprobar con los siguientes comandos, y explicar:

```
x = [-2 -1 1 2]; y=[-6 0 0 6];
l = lagrang_int(x,y);
xx = [-2:0.02:2]; yy = polyval(l,xx);
plot(xx,yy,'b',x,y,'o')
```

d) Analizar y explicar los siguientes comandos:

```
year = [2007 2008 2009 2010 2011 2012];
precio= [133.5 132.2 138.7 141.5 137.6 144.2];
ys = year-mean(year);
l = lagrang_int(ys,precio)
x = linspace(min(year),max(year),200);
ds = x - mean(year);
p=polyval(l,ds); plot(year,precio ,'o',x,p,'-')
```