APUNTES DE MATLAB

Cándido Piñeiro Gómez

Capítulo 1

Comandos Básicos y Matrices

1.1. Introducción

Matlab es un lenguaje de alto nivel orientado al desarrollo de cálculos técnicos. Integra cálculo, visualización y programación en un entorno interactivo de fácil manejo. Los problemas y las soluciones se expresan en la notación matemática habitual.

El elemento básico de información es una matriz a la que no hace falta asignar dimensiones con anterioridad. Por tanto, pueden abordarse problemas que requieren una formulación vectorial o matricial de un modo más fácil que en un lenguaje tipo FORTRAN o C. El nombre MATLAB es una abreviatura de Matrix Laboratory.

1.2. Iniciación a Matlab

Para entrar en Matlab, debemos hacer clic dos veces en el icono correspondiente. Para salir, tenemos dos opciones: 1) en File podemos hacer clic en exit y 2) podemos teclear quit y pulsar enter.

Una vez que hemos entrado en Matlab, escribiremos las instrucciones en la ventana de comandos (command window) a partir del símbolo >>

(denominado prompt). Si se quiere salvar en un archivo toda una sesión de trabajo, podemos proceder como sigue:

Iniciamos la sesión tecleando:

>> diary nombre.txt

y terminar la sesión tecleando

diary off

Automáticamente Matlab crea un archivo con la denominación nombre.txt que se encuentra en el directorio de trabajo. El nombre es opcional, pero un nombre adecuado puede ser poner el mes seguido de la fecha del día en curso: octubre21.txt. Si queremos guardar el archivo en un disquete, CD,.., lo abrimos y en File hacemos clic en save as. Ahora pulsando en Mi PC podemos guardarlo en la unidad que deseemos.

Matlab nos ofrece una ayuda que puede ser de gran utilidad. Si tenemos alguna duda sobre el funcionamiento concreto de determinada función, de la que conocemos su denominación en Matlab, podemos teclear en la ventana de comandos

>> help nombre

y aparece en pantalla la ayuda de Matlab explicando el funcionamiento de la función. Si no fuera suficiente, podemos pulsar en **doc nombre** que nos ofrece una ayuda más completa y ejemplos. Supongamos que tenemos alguna duda sobre el funcionamiento de la función **det** (determinante). Tecleamos

>> help det.

Si tecleamos

>> lookfor nombre

aparece en pantalla una relación con todos los comandos que contienen nombre. Si no queremos que la ayuda aparezca en la ventana de comandos sino en una ventana específica, debemos teclear

>> helpwin nombre

y se abre una ventana con la ayuda.

Finalmente, indicar que también podemos buscar la ayuda acudiendo a Help y haciendo clic en Matlab Help.

1.3. Operaciones básicas y variables

Matlab distingue entre mayúsculas y minúsculas. Entonces A y a pueden ser nombres apropiados para dos variables distintas. El nombre de toda variable debe comenzar por una letra. X, x, X1 o media son nombres válidos de variables.

+, -, *, /, ^ son los símbolos que denotan las operaciones suma, diferencia, producto, cociente y potencia. Las operaciones se van realizando por orden de prioridad, primero las potencias, después las multiplicaciones y divisiones y, finalmente, las sumas y restas. Las operaciones de igual prioridad se llevan a cabo de izquierda a derecha.

Si se quiere asignar a una variable x el valor $3 + 2^5$, se escribe a continuación del prompt

>> x=3+2^5;

Se pueden utilizar las funciones matemáticas usuales. Por ejemplo, si se quiere asignar a la variable a el valor $\sqrt{3}$, se escribe

>> a=sqrt(3);

Los comentarios deben ir precedidos del signo %. Cuando una instrucción no cabe en una línea se puede cortar en dos trozos, en cuyo caso el primero de ellos debe acabar con tres puntos seguidos y el segundo trozo va en la línea siguiente. Para recordar órdenes previas (o posteriores si hemos retrocedido) se usan las teclas de cursor $\uparrow$ y $\downarrow$, y para corregir una instrucción las teclas $\rightarrow$ y $\leftarrow$.

Si se quiere conocer el valor de una variable, basta teclear su nombre. Para conocer las variables que se han usado hasta el momento, se utiliza el comando **who** y, si se quiere más información, **whos**. Para eliminar una variable denominada nombre

>>clear nombre

A continuación damos una relación con los nombres de las funciones más habituales:

raíz cuadrada: sqrt(x).

función exponencial: exp(x).

función seno: sin(x).

función coseno: cos(x).

función tangente: tan(x).

función cotangente: cot(x).

función arcoseno: asin(x).

función arcotangente: atan(x).

función logaritmo neperiano: log(x).

función logaritmo en base 10: log10(x).

Resto de la división entera de m por n: **rem(m,n)**.

Matlab tiene definidas variables con un valor predeterminado. Veamos algunos ejemplos más importantes:

pi. Su valor es el número π.

inf. Su valor es infinito. Aparece, por ejemplo, cuando hacemos 1/0.

eps. Es el número positivo más pequeño con que trabaja Matlab

>>eps

ans 2.2204e-016

Terminamos esta sección explicando cómo puede controlarse el formato numérico con que se muestran en pantalla los resultados numéricos de los cálculos. Sin embargo, hay que señalar que esto no tiene nada que ver con la precisión con la que se realizan dichos cálculos. Si queremos que se muestren en pantalla los resultados con sólo 5 dígitos, se teclea **format short**

>> format short

a=1/3

ans 0.3333

Si deseamos ver en pantalla los resultados con 15 dígitos, se tecleará **format long**. Si tecleamos **format rat**, Matlab busca una aproximación racional.

Ejemplo. Matlab denota el número π por pi. Veamos cuál es el resultado de teclear format rat antes de pi:

>> format rat

pi

ans 355/113.

Finalmente, **format short e** produce el resultado en notación punto flotante con cinco dígitos

Ejemplo.

>> format short e

a=1/3

ans 3.3333e-001

1.4. Vectores

Si queremos introducir las componentes de un vector v, las escribiremos entre corchetes separándolos con comas o espacios.

Ejemplo. El vector fila $v = (1, 2, -1)$ se introduce en Matlab como sigue

>> $v = [1\,2\ -1]$;

Nótese el punto y coma final. Si no se pone, al pulsar enter Matlab muestra en pantalla la fila 1 2 -1. Si se pone el punto y coma, Matlab guarda en memoria el vector $v = (1, 2, -1)$ y no lo muestra en pantalla. Esto es un hecho general que se producirá cada vez que escribimos alguna instrucción. Si colocamos el punto y coma final, Matlab ejecuta la instrucción en cuestión y no muestra en pantalla el resultado ni los cálculos involucrados. Por tanto, es importante tener en cuenta esta característica de Matlab. Por ejemplo, si los cálculos que debe realizar Matlab son numerosos, puede que no nos interese verlos en pantalla.

Si el vector o matriz fila tienen la particularidad de que sus componentes están igualmente espaciadas, hay una forma más simple de introducirlo. Así, el vector $v = (1, 3, 5, 7)$ se puede introducir de la forma siguiente

>> v=1:2:7;

Es decir, se indica, separados por dos puntos, la primera componente, el desfase de uno al siguiente y el último. Cuando el desfase es la unidad se puede omitir. Entonces $v = 4 : 8$, es la forma más simple de indicar el vector $v = (4, 5, 6, 7, 8)$.

Otra forma de introducir un vector fila con las componentes igualmente espaciadas consiste en indicar la primera componente, la última y el número total de componentes.

Ejemplo. El vector v que tiene 10 componentes igualmente espaciadas siendo 1 la primera y 18 la última se indica

$>> v = linspace(1, 18, 10);$

Recuérdese que, si no colocamos al final el punto y coma, aparecerá en pantalla una fila de 10 números que no son otra cosa que las componentes de v.

Cuando necesitemos conocer el número de componentes de un vector v bastará recurrir a la función **length(v)**.

sort(v) es el vector que resulta al escribir las componentes de v de menor a mayor.

Para finalizar, veamos algunas operaciones habituales entre vectores:

- Suma: >> u+v.
- Producto por un escalar: >> a*v.
- Producto escalar: >> dot(u,v).
- Producto vectorial: >> cross(u,v).

1.5. Manipulación de matrices

Para introducir los elementos de una matriz A, escribiremos entre corchetes los elementos de cada fila separados por un espacio (o por una coma), mientras que las filas se separan con punto y coma.

Ejemplo. La matriz

$$A = \begin{pmatrix} 2 & -1 & 0 \\ 3 & 2 & 1 \end{pmatrix},$$

se introduce en Matlab como sigue

>>A=[2 -1 0;3 2 1];

Para referirse al elemento de la fila i y columna j de una matriz A, escribiremos A(i,j), como es usual en la teoría de matrices. También podemos

necesitar referirnos a alguna submatriz de A. Por ejemplo, si necesitamos la fila i de A, escribiremos A(i,:). Del mismo modo, la columna j de A se denota en Matlab por A(:,j). La submatriz de A formada por las filas 2 y 3 y las columnas 1 y 2 de A se denota por A([2 3],[1 2]). Cuando las filas (o las columnas) que forman la submatriz son consecutivas hay una forma más simple de denotarla

>>A(1:4,[2 4])

es la forma de referirse a la submatriz de A formada por las cuatro primeras filas y por las columnas 2 y 4.

Ciertas matrices muy simples, entre ellas la matriz unidad y la matriz nula, no necesitan ser introducidas elemento a elemento ya que Matlab dispone de funciones que nos permitirán referirnos a ellas de forma muy fácil.

1. Matriz unidad: eye(4) es la matriz unidad de orden 4.

2. Matriz nula: zeros(3,4) es la matriz nula 3×4.

3. ones(n,m) es la matriz $n \times m$ cuyos elementos son todos iguales a 1.

4. rand(n,m) es una matriz $n \times m$ de números aleatorios distribuidos uniformemente en el intervalo (0,1). randn(n,m) es una matriz de números aleatorios como antes, pero ahora están distribuidos normalmente.

5. det(A) nos da el determinante de una matriz cuadrada A.

6. rank(A) es el rango de la matriz A.

7. A' es la matriz traspuesta y conjugada de A.

8. inv(A) es la matriz inversa de la matriz cuadrada y regular A.

9. sin(A) es la matriz $(\text{sen}(a_{ij}))$, donde $A = (a_{ij})$. Análogamente ocurre con las otras funciones de Matlab que hemos descrito en una sección anterior.

10. diag(A) es una matriz columna con los elementos de la diagonal principal de A.

11. size(A) nos da las dimensiones de la matriz A.

1.6. Operaciones con matrices

Los símbolos + , * y ^ se reservan para las operaciones matriciales suma, producto y potencia.

Ejemplos. a) Determinar el cuadrado de la matriz cuadrada A:

>> A^2

b) Determinar el producto de las matrices A y B (recordar que el producto sólo es posible si el número de columnas de A es igual al de filas de B:

>> A*B

c) Sumar las matrices A y B (que deberán tener la misma dimensión):

>> A+B

También necesitaremos operaciones como las siguientes: el producto de dos matrices elemento a elemento, elevar cada elemento de una matriz a un cierto exponente, dividir elemento a elemento una matriz por otra. En general, estas operaciones se indicarán anteponiendo un punto al símbolo usado para denotar la operación en cuestión. A continuación damos una relación de estas operaciones:

A.*B es la matriz cuyos elementos se obtienen haciendo el producto de cada elemento de A por el correspondiente de B.

A.^a es la matriz que resulta al elevar cada elemento de A al exponente a.

A./B es la matriz cuyos elementos se obtienen dividiendo cada elemento de A por el correspondiente de B.

1.7. Operaciones entre escalares y matrices

a+B es la matriz que resulta al sumar el escalar a cada elemento de la matriz B.

a*B cada elemento de B se multiplica por el escalar a.

A/b cada elemento de A se divide por el escalar b.

1.8. Las instrucciones A/b y $A\backslash b$

Supongamos que A es una matriz cuadrada de orden n y b un vector columna de n componentes. $A\backslash b$ nos da la solución del sistema de ecuaciones $Ax = b$.

Ejemplo. Resolver el sistema $\begin{cases} x + 2y = 2 \\ 3x + y = 0. \end{cases}$

Escribimos el sistema en forma matricial: $A\binom{x}{y} = b$ donde $b = \binom{2}{0}$ y A es la matriz

$$\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}.$$

Si el determinante de A es no nulo, sabemos que existe la matriz inversa A^{-1}. Entonces, multiplicando la igualdad $A\binom{x}{y} = b$ por A^{-1} (a la izquierda), resulta $\binom{x}{y} = A^{-1} \cdot b$, que es la solución del sistema. Matlab permite encontrar esta solución mediante la instrucción $A\backslash b$.

>> A=[1 2;3 1];

b=[2;0];

$[x; y] = A\backslash b$

ans $\binom{x}{y} = \binom{-0.4}{1.2}$.

Más en general, si B es una matriz con un número de filas igual que el número de columnas de A, $A\backslash B$ nos da la solución del sistema $A \cdot X = B$.

Ejemplo. Resolver los dos sistemas siguientes

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}; \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

Ambos sistemas tienen la misma matriz de coeficientes, A. Matricialmente, los dos sistemas equivalen a la igualdad $A \cdot X = B$, donde

$$X = \begin{pmatrix} x1 & x2 \\ y1 & y2 \\ z1 & z2 \end{pmatrix} \text{ y } B = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{pmatrix}.$$

Podemos resolver los dos sistemas de un tirón procediendo como sigue

>> A=[2 1 1;1 2 1;1 1 2];

B=[1 1;1 0;1 -1];

$A\backslash B$

ans
$$\begin{pmatrix} 0.25 & 1 \\ 0.25 & 0 \\ 0.25 & -1 \end{pmatrix}.$$

Por otra parte, la instrucción B/A nos ofrece la solución del sistema
$X \cdot A = B$**.**

Ejemplo. Resolver el sistema $(xyz) \cdot A = (101)$, siendo A la matriz

$$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 2 & 1 & 1 \end{pmatrix}.$$

>> A=[1 2 1;1 0 -1;2 1 1];

b=[1 0 1];

b/A

ans $(x, y, z) = (-0.5 - 0.51)$.

1.9. Matrices definidas mediante cajas

Cuando una matriz A se compone de cajas que ya hemos introducido, se puede introcucir A de una manera más cómoda como muestra el ejemplo siguiente.

Ejemplo. Supongamos que ya hemos introducido las matrices:

$$B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}; C = \begin{pmatrix} -1 & 3 \\ 4 & 1 \end{pmatrix},$$

entonces la matriz

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 2 & 2 & 1 & -1 & 3 \\ 2 & 2 & 2 & 4 & 1 \end{pmatrix},$$

se puede introducir como sigue

>>A=[zeros(2,3) eye(2,2);2*ones(2,2) B C]

Siempre que definamos una matriz por cajas deberemos tener en cuenta las dos restricciones siguientes:

a) En cada nivel el número de filas de las cajas es el mismo.

b) En todos los niveles el número total de columnas es el mismo.

Capítulo 2

Gráficos

2.1. Curvas planas

Si se desea obtener la gráfica de la función $y = y(x)$ en el intervalo [a,b], debemos tener presente que Matlab dibuja las curvas punto a punto; es decir, calcula los puntos $(x, y(x))$, para los valores de x que le indiquemos y representa dichos puntos unidos por un segmento. Por ello, se empieza estableciendo la matriz fila x cuyos elementos son los valores de x para los que se computará el valor correspondiente de $y(x)$. Lo usual será tomar puntos igualmente espaciados en el intervalo [a,b], incluyendo los extremos. Tomando la distancia entre dos valores consecutivos de x convenientemente pequeña, el aspecto final será el de una verdadera curva en lugar de una poligonal.

Ejemplo . *Dibujar la curva de ecuación* $y = xsenx$ *en el intervalo* $[-2\pi, 2\pi]$.

```
>> x=linspace(-2*pi,2*pi,60);          % Tomamos 60 valores de x igual
                                       % mente espaciados en [−2π, 2π]
y=x.^ 2.*sin(x);% matriz fila con los valores de y(x)
plot(x,y)
```

Pulsando enter, se abre una ventana gráfica con la curva. Se pueden dibujar varias curvas en la misma ventana gráfica. Si el intervalo de variación de x es el mismo, se puede proceder como se muestra en el ejemplo siguiente.

Ejemplo.*Representar gráficamente las curvas* $y1 = x^2$ *e* $y2 = xe^x$ *en el*

intervalo [-3,3].

```
>> x=linspace(-3,3,90);
y1=x.^ 2;y2=x.*exp(x);
plot(x,y1,x,y2)
```

De esta forma se consigue que se abra una ventana gráfica con las dos curvas. **Otra forma de conseguir el mismo resultado consiste en usar la orden hold on. Si ya tenemos una ventana gráfica con una curva y queremos dibujar una segunda curva en la misma ventana, ponemos hold on y a continuación las órdenes necesarias para dibujar la segunda curva.**

Si el intervalo donde se quiera dibujar cada curva no es el mismo, se puede conseguir el mismo resultado de la forma siguiente. Se dibuja primero una de las curvas, se teclea hold on y acto seguido se dibuja la otra

Ejemplo. Dibujar $y = x$ en [-1,1] e $y = xe^x$ en [0,2].

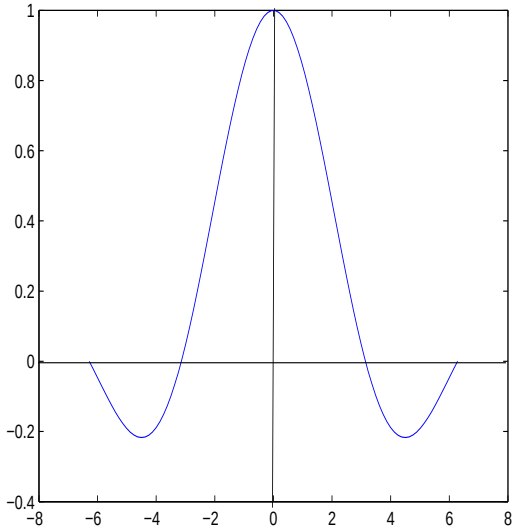
```
>> x1=-1:.1:1;
y1=x1;
plot(x1,y1)
hold on
x2=0:.1:2; y2=x2.*exp(x2);
plot(x2,y2)
```

Por el contrario, cuando ya se tiene una ventana gráfica abierta y se quiere dibujar una nueva curva en otra ventana gráfica, pero sin perder la primera ventana, tecleamos figure y se abre una ventana gráfica nueva donde podremos hacer la nueva representación.

EJEMPLO. Supongamos que se desea obtener la gráfica de $f(x) = \text{sen}(x)/x$, para $x \in [-2\pi, 2\pi]$. Se presenta el problema de que para $x = 0$ no está definida la función, aunque $\lim_{x\to 0} f(x)$ existe y vale 1. Manejando adecuadamente la variable eps, podemos evitar este problema.

```
>>x=linspace(-2*pi,2*pi,100);
t=x+eps;
y=sin(t)./t;
```

plot(t,y)
y obtenemos



Otras opciones de la función plot

1. Etiquetas sobre los ejes: el comando xlabel('texto') se usa para que en el eje OX aparezca el *texto* que se desea. Análogamente, ylabel('texto') para el eje OY. Para colocar cualquier cadena de texto en el punto que queramos de la ventana gráfica, se usa el comando text(x,y,'texto'), donde (x,y) son las coordenadas del punto donde queremos situar el centro izquierda del texto. Si usamos el comando gtext, entonces podemos colocar el texto donde queramos con el propio ratón.

2. Color de la curva y estilo de línea: Se puede dibujar la curva del color y con el estilo que se desee. Por ejemplo, si se escribe

>>plot(x,y,'.',x,z,'- -') De esta forma se consigue que la curva $y = y(x)$ aparezca con trazo punteado y la $z = z(x)$ como una línea de trazo discontinuo.

Si se quiere escoger el color y estilo:

>> plot(x,y,'r - -')

la curva aparece en rojo y con línea de trazo discontinuo. La siguiente tabla muestra la sintaxis de los diferentes colores y tipo de línea:

Símbolo	Color	Símbolo	Estilo de línea
y	amarillo	.	línea de puntos
m	magenta	$\circ$	círculo
r	rojo	$+$	más
g	verde	$*$	estrella
b	azul	$--$	trazo discontinuo
k	negro	$-$	línea sólida

Ejes a medida

Para fijar los valores máximo y mínimo de los ejes:

>>axis([xmin xmax ymin ymax])

Para que la escala sea la misma en ambos ejes:

>>axis equal o axis('equal')

Para que la gráfica sea un cuadrado:

>>axis square

Si se quiere que aparezca una rejilla: grid on.

Dibujo de poligonales

Supongamos que se desea dibujar la poligonal de vértices (x_i, y_i) con $i = 1, .., n$. Definiríamos las matrices fila x e y que contienen las coordenadas correspondientes y el comando plot(x,y) dibuja la poliginal (recordar cómo dibuja las curvas Matlab). Si la poligonal es cerrada, el último vértice ha de ser (x_1, y_1).

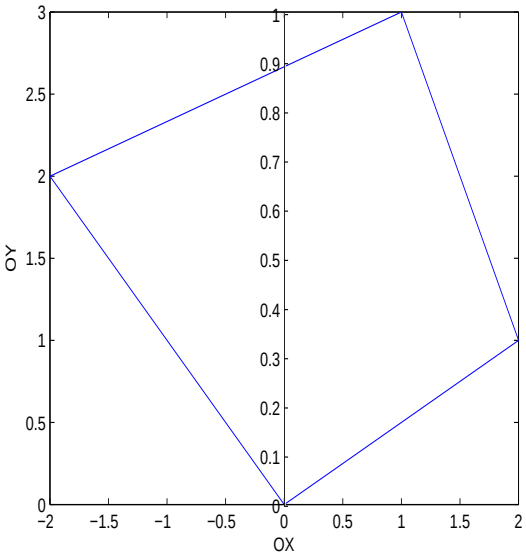
Ejemplo. *Dibujar la poligonal cerrada de vértices* $(0, 0), (2, 1), (1, 3)$ y $(-2, 2)$.

>>x=[0 2 1 -2 0];

y=[0 1 3 2 0];

plot(x,y)

y el resultado sería



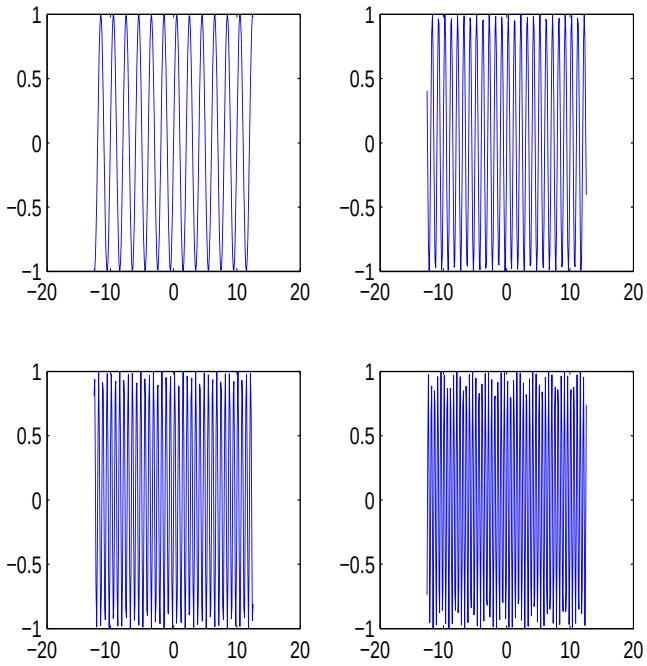
2.2. El comando subplot

A veces nos interesará disponer en una misma ventana gráfica de varias subventanas para dibujar en cada una de ellas una curva distinta, con el objetivo de poder compararlas más cómodamente. Veamos un ejemplo.

Ejemplo. Queremos dibujar en una misma ventana gráfica las curvas $y = \text{sen}(k\pi x)$, para $k = 1, ..., 4$ y $x \in [-4\pi, 4\pi]$, cada una en una subventana diferente.

```
>>x=linspace(-4*pi,4*pi,240);
subplot(2,2,1)
plot(x,sin(pi*x))
subplot(2,2,2)
plot(x,sin(2*pi*x))
subplot(2,2,3)
plot(x,sin(3*pi*x))
subplot(2,2,4)
plot(x,sin(4*pi*x))
```

Y se obtiene



En general, si se necesitan $m \times n$ subventanas, se tendrá en cuenta que se numeran de izquierda a derecha y de arriba hacia abajo. Cuando tecleamos subplot(m,n,k), estamos indicando que vamos a dibujar en la subventana que ocupa el lugar k. En el ejemplo anterior, todas las subventanas responden a la forma subplot(2,2,k), porque teníamos 4 curvas y parece lo más adecuado disponerlas en dos filas y dos columnas.

2.3. Coloreado de polígonos

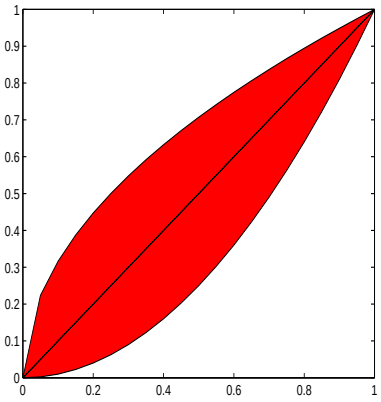
En la sección anterior hemos visto cómo se dibuja una poligonal con Matlab. Ahora vamos a ver cómo se colorea la región interior del color que queramos.

Ejemplo. Se desea dibujar las curvas $y = x^2$ y $x = y^2$ en el primer cuadrante. La región que encierran debe aparecer en color rojo.

```
>> x=0:.1:1;
y1=x.^2;y2=sqrt(x);
plot(x,y1,x,y2)
```

De esta forma se han dibujado las curvas en cuestión. Para colorear de rojo la región encerrada, Matlab dispone de la función fill. Su sintaxis es la siguiente: fill(x,y,'r') (x e y son matrices fila que contienen las coordenadas x e y de los vértices de la poligonal. El programa anterior se continuaría de la forma siguiente:

```
X=[x x];
y=[y1 y2];
fill(X,y,'r')
```



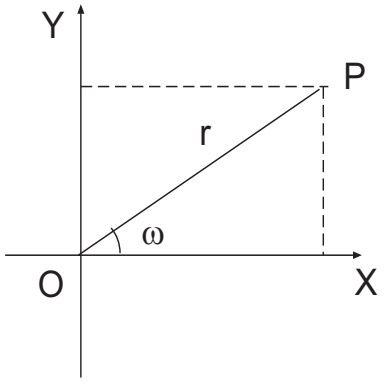
Si se desea escoger un color indicando las coordenadas (en el sistema RGB) se escribe

```
>>fill(x,y,[r g b])
```

Los números r, g y b pertenencen a (0,1) y representan las proporciones en que se deben tomar los colores principales (rojo, verde y azul) para crear el color en cuestión.

2.4. Curvas en polares

Empezamos recordando cómo se relacionan las coordenadas polares con las cartesianas.



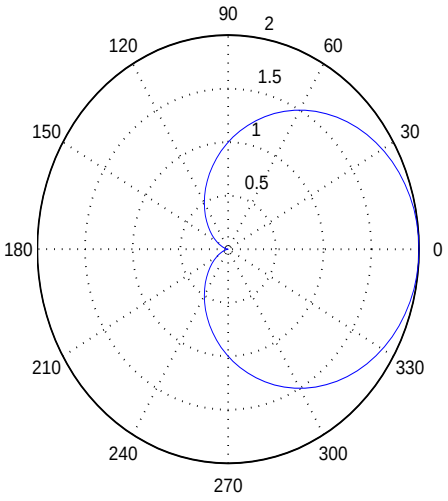
Vemos en la figura que ω es el ángulo que forman el vector de posición del punto P con la dirección positiva del eje OX y r es el módulo de dicho vector. Por un lado, el Teorema de Pitágoras nos dice que $r^2 = x^2 + y^2$ y, por otro, usando las definiciones de $\operatorname{sen}\omega$ y $\cos\omega$, obtenemos $x = r\cos\omega$ e $y = r\operatorname{sen}\omega$. Si de una curva plana sabemos que las coordenadas polares de sus puntos, (r, ω), verifican la igualdad $r = r(\omega)$, para $\omega \in [\omega_1, \omega_2]$, diremos que $r = r(\omega)$ es la ecuación de la curva en coordenadas polares. La ecuación de la circunferencia unidad en cartesianas es $x^2 + y^2 = 1$ y en coordenadas polares $r = 1$. En general, para obtener la ecuación en polares, conocida la ecuación de una curva en cartesianas, basta sustituir en esta última ecuación x e y por $r\cos\omega$ y $r\operatorname{sen}\omega$, respectivamente.

Ejemplo. *Dibujar la curva de ecuación* $r = 1 + \cos\omega$*, para* $0 \leq \omega \leq 2\pi$.

Por comodidad, vamos a usar w en lugar de ω.

```
>> w=linspace(0,2*pi,60);
r=1+cos(w);
polar(w,r)
```

pulsando enter se abre una ventana gráfica que muestra la curva siguiente (denominada cardioide).



2.5. Curvas en el espacio

Supongamos que se quiere dibujar la curva de ecuaciones paramétricas $x = \cos t, y = \operatorname{sen} t, z = t$, para $t \in [0, 6\pi]$. Podemos usar el comando plot3 o el comando ezplot3.

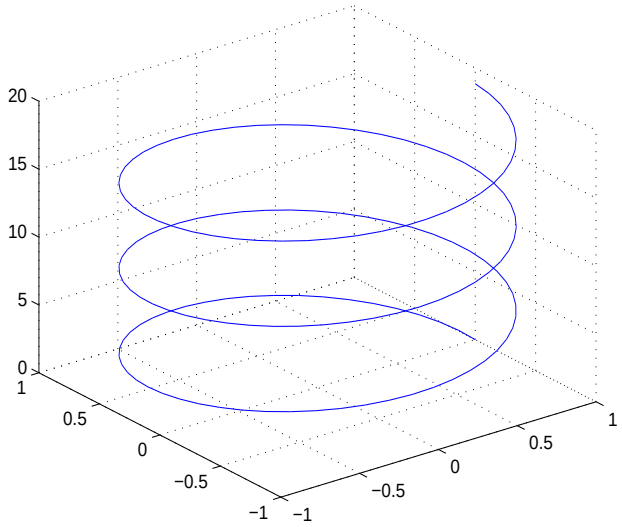
a) Con plot3:

>>t=linspace(0,6*pi,150);

plot3(cos(t),sin(t),t)

grid on

y el resultado es



b) Con ezplot3:

ezplot3('cos(t)','sin(t)','t', [0,6*pi])

Terminamos esta sección indicando cómo puede conseguirse que aparezcan los vectores tangentes al dibujan curvas en paramétricas .

Ejemplo. Representar la curva de ecuaciones paramétricas

$$x = \cos(t),\ y = \text{sen}(t),\ t \in [0, pi].$$

>>t=linspace(0,pi,30);

plot(cos(t),sin(t))

Si se quiere que aparezcan los vectores tangente, se usa la función **quiver**.

>>t=linspace(0,pi,30);

plot(cos(t),sin(t))

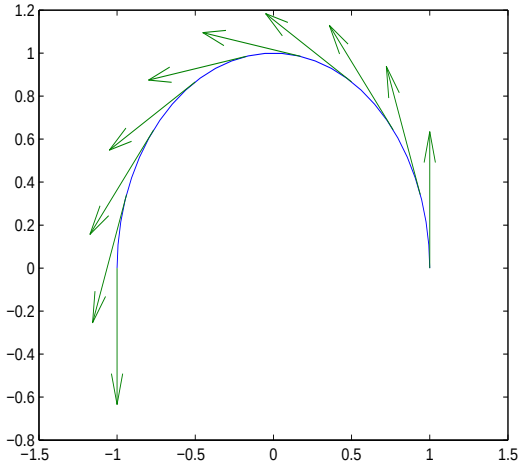
hold on

>> t=linspace(0,pi,10);%Dibujamos el vector tangente en sólo 10 puntos

% intermedios de la curva

quiver(cos(t),sin(t),-sin(t),cos(t))

y se obtiene



2.6. Superficies

Para dibujar una superficie de ecuación $z = z(x, y)$, se comienza por establecer los intervalos de variación de x e y. Con la orden [x,y]=meshgrid(-2:.1:2;-1:.1:1) Matlab crea una matriz x con todas sus filas iguales a -2:.1:2 y una matriz y con todas sus columnas iguales a -1:.1:1. De este modo resultan dos matrices con la misma dimensión. Veamos un ejemplo simple. Consideramos la función $z = x^2 + y^2$ y escribimos la orden

>> [x,y]=meshgrid(-1:.5:1,0:.5:1);

que produce las matrices

$$x = \begin{pmatrix} -1 & -.5 & 0 & .5 & 1 \\ -1 & -.5 & 0 & .5 & 1 \\ -1 & -.5 & 0 & .5 & 1 \end{pmatrix}, \quad y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ .5 & .5 & .5 & .5 & .5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

De esta forma, cuando escribamos z=x.^2+y.^2 en la ventana de comandos, z será la matriz 3×5 que contiene los valores de z en todos y cada uno de los puntos (x, y) con x igual a uno de los valores -1,-.5,0,.5,1 e y igual a uno de los valores 0,.5,1.

Ejemplo. *Dibujar la superficie* $z = \sqrt{x^2 + y^2}$ *en el dominio* $[-3, 3]\times[-3, 3]$

>>[x,y]=meshgrid(-3:.1:3,-3:.1:3);

z=sqrt(x.^2+y.^2);

surf(x,y,z)

Además de la orden surf(x,y,z), para dibujar una superficie, podemos emplear plot3(x,y,z) o mesh(x,y,z) (la diferencia con surf es que ésta rellena los espacios entre líneas).

La orden rotate3d permite girar la superficie con el ratón. La orden colormap permite cambiar el color de la superficie. Hay diversas opciones: colormap(pink), colormap(summer),colormap(winter), etc.

En ele ejemplo siguiente mostramos cómo se dibuja una superficie plana paralela a $z = 0$.

Ejemplo. Dibujar la superficie $z = 3$ en el primer octante.

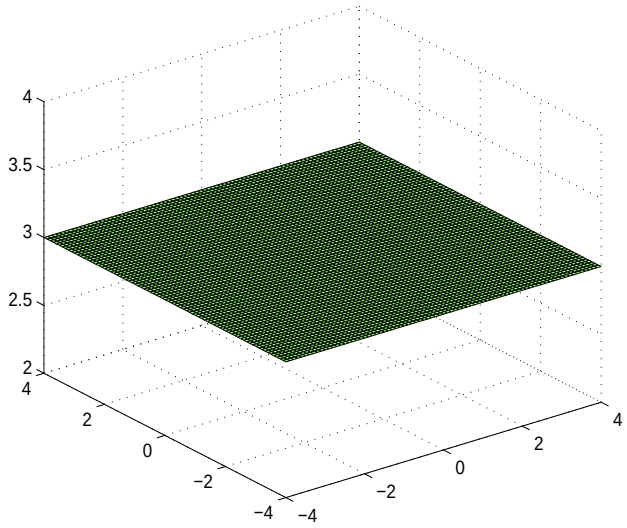
>> [x,y]=meshgrid(-4:.1:4,-4:.1:4);

[m,n]=size(x); % Recordar lo que hemos dicho sobre el comando meshgrid,

% usamos size(x) para determinar las dimensiones de x de una manera segura.

z=3*ones(m,n);

surf(x,y,z)



2.7. Curvas de nivel

Supongamos que hemos dibujado una superficie, $z = f(x, y)$, y queremos que en el plano $z = 0$ aparezcan dibujadas las curvas de nivel. Podemos proceder como sigue

>>hold on

contour(x,y,z,[c1,c2,c3,...,cn])

y aparecen representadas las curvas de nivel $f(x, y) = ci$, para $i = 1, .., n$.

Ejemplo. Representar la superficie de ecuación $z = x^2 + y^2$, para $(x, y) \in [-2, 2] \times [-2, 2]$. Además, deseamos que aparezcan representadas las curvas de nivel $f(x, y) = c$, para $c = 1, 2, 3$.

>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);

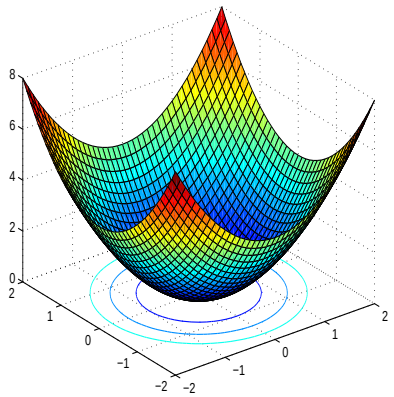
z=x.^2+y.^2;

surf(x,y,z)

hold on

contour(x,y,z,[1,2,3])

y se obtiene



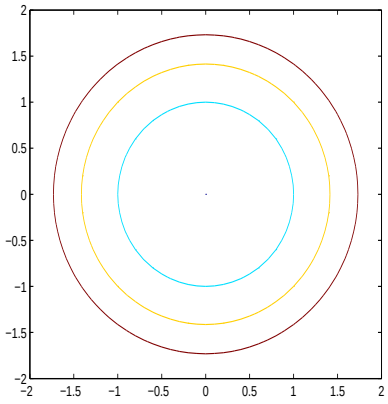
Si sólo se buscan las curvas de nivel (no se necesita dibujar la superficie), basta escribir

>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);

z=x.^2+y.^2;

contour(x,y,z,[1,2,3])

y se obtiene



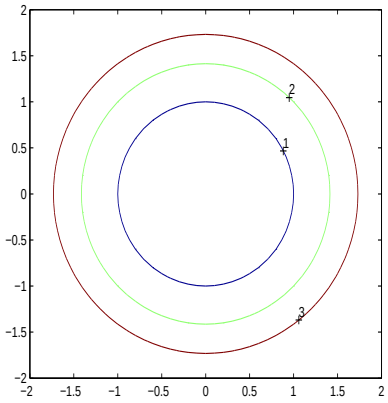
Si se quiere que aparezcan etiquetas sobre cada curva de nivel que reflejen el valor de la constante c, se usa la función **clabel**

>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);

z=x.^2+y.^2;

clabel(contour(x,y,z,[1,2,3]))

resultando



Cuando necesitamos representar una superficie y deseamos que aparezcan las curvas de nivel, hay una forma muy cómoda que consiste en emplear la función **surfc(x,y,z)**.

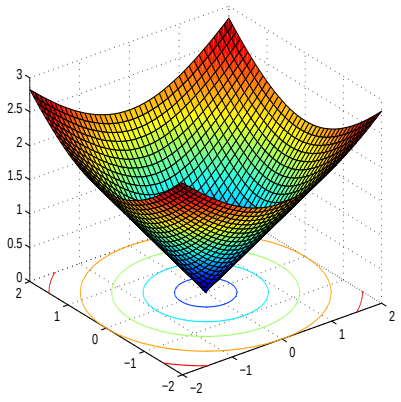
Ejemplo. Gráfica de la superficie $z = \sqrt{x^2 + y^2}$ con las corrspondientes curvas de nivel.

>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);

z=sqrt(x.^2+y.^2);

surfc(x,y,z)

y resulta



2.8. ¿Cómo exportar gráficos?

a) Word. Una vez que tenemos el gráfico en la ventana gráfica, vamos a Edit (en le ventana gráfica) y hacemos clic en Copy figure. A continuación abrimos el documento word y pegamos.

b) Latex. Primero debemos salvar el gráfico de forma adecuada. Vamos a File (en la ventana gráfica) y hacemos clic en Save as. Se abre una ventana (Save as) en la que debemos dar el nombre con que vamos a designar el archivo que contendrá el gráfico. Automáticamente, Matlab le añade la extensión que corresponda. Normalmente, elegiremos el tipo EPS file, en cuyo caso la extensión que Matlab añade es eps. Ahora es el momento de guardar el

archivo creado, nombre.eps, en la carpeta que contiene el archivo Latex donde queremos incluir el gráfico. Ahora debemos preparar el archivo Latex, lo que exige realizar los siguientes dos pasos

1) En el archivo principal y antes de la instrucción \begin{document}, debemos incluir

\usepackage{graphicx}

2) En el lugar donde vamos a incluir el gráfico:

\begin{center}

\includegraphics[width=5cm,height=6cm]{nombre.eps}

\end{center}

Capítulo 3

Programación

3.1. Archivos .M

Vamos a ver que podemos crear dos tipos de archivos con extensión .m, que llamaremos archivos .M. Se denominan archivos de función y archivos de guión (o de instrucciones). Con los primeros podemos definir nuevas funciones que se añadirán a las que ya trae Matlab (como sin(x), exp(x),sqrt(x),etc.). Los archivos de instrucciones se llaman así porque pueden consistir en una serie de instrucciones a ejecutar. Veremos que en éstos puede ocurrir que en el momento de su ejecución nos pida una serie de valores (inputs). En todos los casos, se deberá ir a File-New-M-File y aparece una ventana (Editor-Untitled) donde podemos escribir el programa correspondiente. Una vez terminado, vamos a File de dicha ventana y hacemos clic en Save As y podemos guardar el archivo .M con el nombre que le hayamos dado (nombre.m).

3.2. Archivos de función

Las dos primeras líneas de un archivo de función tienen la forma
function [y,z,..]=nombre(a,b,..)

% Una explicación que sirva para reconocer la función en cualquier otro

momento

a,b,.. denotan las variables de entrada (las variables independientes), mientras que y,z,.. son las variables de salida (dependientes), en ambos casos separadas por comas. A continuación van todas las órdenes que se necesitan para definir la nueva función.

Ejemplo. *Crear un archivo de función cuya entrada sea una matriz fila x y cuyas salidas sean la media de x y su desviación típica.*

```
function [media,dest]=estadisticos(x)
% Esta función determina la media y la desviación típica de una fila x
n=length(x);
media=sum(x)/n;
s=0;
for k=1:n
s=s+(x(k)-media)^2;
end
dest=sqrt(s/n);
[media,dest]
```

El nombre de todo archivo de función debe coincidir con el nombre de la función y tiene extensión .m. En nuestro caso, sería: nombre.m. El nombre de la función debe empezar con una letra y, para evitar confusiones, debemos asegurarnos que no coincide con el nombre de alguna de las funciones de que dispone Matlab.

3.3. Cálculo de integrales usando un archivo de función

Vamos a ver una forma de calcular integrales simples y reiteradas creando un archivo de función con el integrando. Este método nos servirá, además, para resolver otro tipo de problemas que tienen en común que necesitamos hacer manipulable una función.

El primer paso consiste en crear un archivo de función con una función que iremos cambiando en cada problema concreto. Este archivo puede llamarse f.m.

Ejemplo. Deseamos calcular $\int_0^1 e^{x^2}\,dx$.

Creamos el archivo f.m:

function y=f(x)

%y=F(x) es el integrando de una integral simple

y=exp(x.^2);

Ahora usamos la función quad para calcular una integral simple.

>>integrando=@f;% Hacemos manipulable la función

quad(integrando,0,1)

ans 1.4627

Veamos ahora cómo se determina una integral reiterada con la función **dblquad**.

Ejemplo. Calcular $\int_0^1 \Big[\int_{-1}^1 xy\,dx\Big]dy$.

En primer lugar, creamos un archivo de función con el integrando:

function z=fun(x,y)

% z=FUN(x,y) es el integrando de una integral reiterada

z=x.*y;

Finalmente, usamos la función dblquad:

>>integrando=@fun;

a=dblquad(integrando,$x_{min}, x_{max}, y_{min}, y_{max}$)

ans 1.2309e-017

En nuestro caso, $x_{min} = -1$, $x_{max} = 1$, $y_{min} = 0$ e $y_{max} = 1$.

Finalizamos esta sección indicando que si necesitamos evaluar $f(a)$ y disponemos de un archivo de función f.m con la función $y = f(x)$, podemos usar la función **feval**:

g=@f;

feval(g,a)

y, pulsando enter, obtenemos el valor $f(a)$.

3.4. La función fplot

Para dibujar la curva $y = y(x)$ con el comando fplot, debemos crear un fichero de función con la función $y(x)$. Supongamos que a este fichero le hemos llamado fun.m. Para manipular la función, pondremos f=@fun. Veamos esto con un ejemplo.

Ejemplo. *Usar fplot para dibujar la curva* $y = x^2 \operatorname{sen} x$

Empezamos creando un archivo de función para $y(x)$:

function y=fun(x)

y=x.^2.*sin(x);

Una vez creado este archivo con la función, procedemos a dibujar la curva $y = y(x)$ con fplot

>>f=@fun;

fplot(f,[xmin,xmax])

grid on

Si se quiere que la gráfica recoja con mayor detalle el intervalo [ymin,ymax]

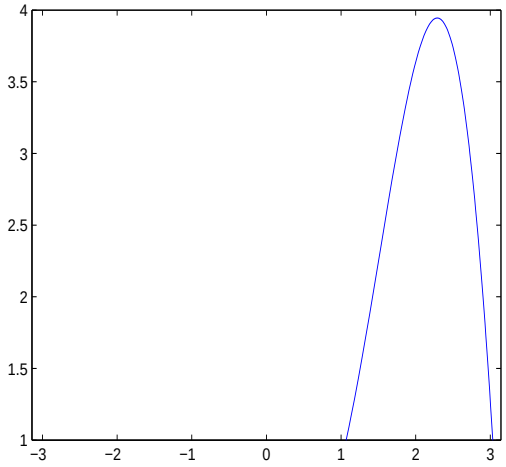
>>fplot(f,[xmin,xmax ymin ymax])

Ejemplo.*Dibujar la curva* $y = x^2 \operatorname{sen} x$ *en* $[-\pi, \pi]$ *con un zoom en [1,4].*

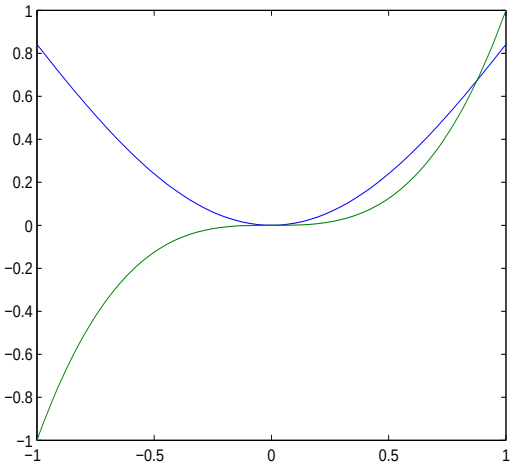
>>f=@fun;

fplot(f,[-pi,pi,1,4])

Al pulsar enter, obtenemos



Otra forma de manejar la función fplot es la siguiente
>>fplot('x*sin(x)',[0,2])
Con la función fplot también se pueden dibujar dos curvas a la vez
>>fplot('[x*sin(x),x^3]',[-1,1])
y resulta



3.5. Archivos de instrucciones

En Programación nos encontramos a menudo con la necesidad de ejecutar varias veces una misma serie de instrucciones. Por ello, puede resultar conveniente crear un archivo de guión con dichas instrucciones, al que podremos recurrir cada vez que lo necesitemos. El nombre de uno de tales archivos sólo tiene la restricción, como ocurre con los archivos de función, de que debe comenzar por una letra. La estructura de estos archivos es la siguiente:

Una primera linea explicativa que describa brevemente el objetivo del archivo. Esta línea de comentarios debe comenzar con %. De esta forma se consigue que el ordenador no considere esta línea.

A continuación se escriben todas las instrucciones que deben ejecutarse en el orden que corresponda.

Ejemplo. Vamos a crear un archivo de guión (que llamaremos raices) y que nos pedirá los coeficientes de una ecuación de segundo grado y determinará las raíces reales cuando las haya. Usaremos el comando **if** que se estudiará con más detenimiento más adelante.

% encuentra las raíces de una ecuación de segundo grado $ax^2+bx+c=0$ y nos pide a,b y c

```
a=input('dame a');
b=input('dame b');
c=input('dame c');
Delta=b^2-4*a*c;
if Delta>=0
[x1,x2]=[-b/(2*a)+sqrt(Delta)/(2*a),-b/(2*a)-sqrt(Delta)/(2*a)]
end
```

Este archivo funciona de la siguiente forma: Cuando tecleamos raices el ordenador nos pide sucesivamente los valores de a, b y c. Calcula el discriminante de la ecuación, Delta, y, al encontrarse con el comando if, si Delta no es negativo, procede a calcular las raíces. Si Delta es negativo, no nos da ningún resultado, pues no ejecuta las instrucciones que hay entre if y end.

3.6. Subfunciones

Un archivo de función puede contener el código de más de una función. La primera función (la función principal) que aparece en el archivo es la que da el nombre a éste y las otras se llamarán subfunciones (sólo son visibles para la función principal o para otra subfunción en el mismo archivo.

Ejemplo. Un archivo de función que determina la media y la mediana de un vector fila.

```
function [media,mediana]=estadistico(v)
% Estadistico encuentra la media y la mediana con subfunciones internas
n=length(v);
media=med(v,n);
mediana=medn(v,n);
function a=med(v,n)
% Calcula la media
a=sum(v)/n;
function b=medn(v,n)
%Calcula la mediana
w=sort(v);
if rem(n,2)==1
b=w((n+1)/2);
else
b=(w(n/2)+w((n/2)+1))/2;
end
```

3.7. Operadores y Funciones lógicas

A) **OPERADORES**. Matlab posee operadores de tres clases: aritméticos, de relación y lógicos.

1) Operadores aritméticos. Realizan computaciones numéricas (sumas, productos, etc.)

2) Operadores de relación. Comparan cuantitativamente dos expresiones. Matlab posee los siguientes:

$<$ menor que

$<=$ menor o igual que

$>$ mayor que

$>=$ mayor o igual que

$==$ igual que

$\sim=$ no igual a

Si se usa uno de estos operadores para comparar dos matrices o vectores, entonces la comparación se hace elemento a elemento. Si se emplean para comparar un escalar con un vector o matriz, entonces se compara el escalar con cada elemento del vector o matriz.

Ejemplo. Consideremos las matrices

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -1 \\ 2 & 1 & 2 \end{pmatrix}, B = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 1 & 4 \\ 1 & 0 & 2 \end{pmatrix}.$$

si ponemos

>>A==B

produce la matriz

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

es decir, la matriz con un 1 en las posiciones donde los elementos de ambas matrices coinciden y un 0 en los restantes.

Operadores lógicos. Matlab emplea los tres operadores lógicos siguientes: $\&, |$ y $\sim$. Cuando operan sobre una matriz, lo hacen elemento a elemento. Cualquier número no nulo se considera como verdadero y el cero como falso. Consideremos, por ejemplo, la instrucciones siguientes:

>>A=[1 0;2 1];

B=[4 0;1 3];

C=A& B

ans $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

Es decir, produce una matriz con unos y ceros , según que los elementos correspondientes de A y B sean no nulos o uno de ellos, al menos, nulo, respectivamente. Si escribimos $A(C)$, Matlab produce una columna con los elementos de A que se corresponden con los unos de C (por columnas, de arriba abajo y de izquierda a derecha). Concretamente, obtenemos

$$A(C) = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

B) **FUNCIONES LÓGICAS.**

find. Esta función determina los índices de una matriz que verifican determinada expresión lógica. Es muy útil para crear filtros o matrices de índices.

Ejemplo. Sustituir en la matriz A = magic(4) los elementos mayores que 8 por 100.

>>A=magic(4)

ans

$$\begin{pmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{pmatrix}$$

i=find(A>8);

A(i)=100

A

ans $\begin{pmatrix} 100 & 2 & 3 & 100 \\ 5 & 100 & 100 & 8 \\ 100 & 7 & 6 & 100 \\ 4 & 100 & 100 & 1 \end{pmatrix}$.

Si se suprime el punto y coma al final de la línea i=find(A>8), en pantalla se despliega el valor de i y vemos que se trata de una matriz columna con los

índices de los elementos de A que superan a 8. Los índices de A se numeran por columnas de izquierda a derecha y de arriba abajo.

all. Si x es un vector con todas sus componentes no nulas, all(x) es igual a 1; en otro caso su valor es cero. Si A es una matriz, all(A) es una matriz fila de unos y ceros, que no es otra cosa que el resultado de aplicar all a cada columna de A (de izquierda a derecha).

any. Si x es un vector con alguna componente no nula, entonces any(x) produce un uno. Por el contrario, si x es nulo, el valor de any(x) también es cero. Sobre una matriz actúa como en el caso de all.

xor. Si x e y son vectores, xor(x,y) es un vector fila de ceros y unos, obtenido de la siguiente forma. Si en determinada posición, sólo uno de los dos vectores tiene componente no nula, entonces xor(x,y) tiene un uno en dicha posición. En cualquier otro caso, tiene un cero.

3.8. Control de flujo

Normalmente, Matlab va realizando las instrucciones de un programa en el orden que las hemos escrito. No obstante, hay varias formas de conseguir que este orden no se respete:

- If, junto con else y elseif, ejecuta un grupo de instrucciones dependiendo de que cierta expresión lógica sea cierta o no.

- While ejecuta un grupo de instrucciones un número indefinido de veces que depende de que cierta expresión lógica se verifique o no.

- For ejecuta un grupo de instrucciones un número establecido de veces.

- Break termina la ejecución de un for o while.

- Switch, junto con case y otherwise, ejecuta diferentes grupos de instrucciones dependiendo del valor de alguna condición lógica.

- Continue pasa el control a la siguiente iteración de un for o while, ignorando cualquier instrucción posterior (del for o while en cuestión).

3.9. If, else y elseif

If evalúa una expresión lógica y ejecuta un grupo de instrucciones si la expresión lógica es verdadera. La forma más simple es la siguiente

>> if expresión lógica

instrucciones

end

Ejemplo. Supongamos que queremos obtener las raíces de una ecuación de segundo grado $ax^2 + bx + c = 0$.

Podemos crear un archivo de instrucciones que nos pida los coeficientes a, b y c y determine las raíces, una vez que haya comprobado que son reales. Le daremos el nombre raices.m.

>> % raices determina las raíces de $ax^2 + bx + c = 0$, nos pide los coeficientes

a=input('dame coefic de xˆ');

b=input('dame coefic. de x');

c=input('dame ter.indep');

Delta = bˆ2-4*a*c;

if Delta<0

disp('no hay raíces reales')

end

[x1,x2]=[(-b+sqrt(Delta))/2*a,(-b-sqrt(Delta))/2*a]

Para entender cómo funciona elseif y else, consideramos el mismo ejemplo anterior, pero vamos a establecer distintas salidas, según sea el valor de Delta.

>> % raices1 determina las raíces de $ax^2 + bx + c = 0$, nos pide los coefi-

% cientes

a=input('dame coefic de xˆ');

b=input('dame coefic. de x');

c=input('dame ter.indep');

Delta = bˆ2-4*a*c;

if Delta<0

disp('no hay raíces reales')

elseif Delta>0
[x1,x2]=[(-b+sqrt(Delta))/2*a,(-b-sqrt(Delta))/2*a]
else
x=-b/(2*a)
disp('raíz doble')
end

elseif evalúa la expresión lógica que aparece en su misma línea si if o los anteriores elseif resultan falsos. Si dicha expresión lógica es verdadera se ejecutan las instrucciones siguientes hasta el próximo elseif o else.

Por el contrario, else no lleva expresión lógica a evaluar y matlab realiza las instrucciones siguientes hasta end si las expresiones lógicas de if y los elseif anteriores so falsos. La forma general de un if es la siguiente:

>>if expresión lógica
instrucciones
elseif expresión lógica
instrucciones
elseif expresión lógica
instrucciones
————-
————
elseif expresión lógica
instrucciones
else
instrucciones
end

3.10. For

La sintaxis es la siguiente:
>>for índice=inicio:incremento:final
instrucciones

end

Si el incremento es la unidad, no hace falta indicarlo.

Ejemplo. Crear un archivo de función para determinar $n!$.

Vamos a File-New para abrir un nuevo archivo .M y escribimos

function y=factorial(n)

% Esta función determina el factorial de un natural N

y=1;

for k=1:n

y=y*k

end

Ejemplo. Construir una matriz A, $n \times m$, tal que $a_{ij} = 1/(i + j - 1)$.

for i=1:n

for j=1:m

A(i,j)=1/(i+j-1);

end

end

3.11. while

Su sintaxis es la siguiente:

>>while expresión

instrucciones

end

Mientras que la expresión que controla el while sea verdadera, se ejecutan todas las instrucciones comprendidas entre while y end.

Ejemplo. Calcular el primer natural n tal que $n!$ es un número con 100 dígitos.

>>n=1;

while prod(1:n)<1e100

n=n+1;

end

3.12. Break

Esta instrucción se usa para terminar un for o un while antes de que se haya completado la ejecución. Si éste no es el caso, se usará el comando return.

Ejemplo. Supongamos que se desea determinar un vector v con un número de componentes menor o igual que 10 y que verifique: a) sus componentes son de la forma 24-kˆ2 y b) son no negativas.

```
>>for k=1:10
w(k)=24-kˆ2;
if w(k)<0
break
end
end
```

El programa anterior determina un vector de la forma $w = (23, 20, 15, 8, -1)$. Es decir, el for se termina cuando $k = 5$ y $w(5) = -1$, pues la expresión lógica del if, $w(k) < 0$ es cierta y Matlab ejecuta la instrucción break que termina el for, aunque no se ha llegado al final ($k = 10$). El vector buscado sería

```
>>n=length(w);
v=w(1:n-1);
```

Es decir, basta eliminar la última componente de w.

3.13. Continue

Este comando pasa el control a la siguiente iteración en un for o un while, sin realizar las instrucciones siguientes que pueda haber antes de end.

Ejemplo. Determinar un vector y cuyas componentes sean de la forma $\operatorname{sen} x/x$, para $x = -1 : .1 : 1$, exceptuando $x = 0$.

```
>>k=0;
for x=-1:.1:1
if x==0
```

continue
end
k=k+1;
y(k)=sin(x)/x;
end

3.14. Switch

La sintaxis es como sigue
switch expresión %(o variable)
case valor1
instrucciones %(se ejecutan si el valor de expresión es valor1
case valor2
instrucciones %(se ejecutan si el valor de expresión es valor2)
———-
otherwise
instrucciones %(se ejecutan si el valor de expresión no es ninguno de los valores de cada caso)
end

Ejemplo. Supongamos que una variable escalar se llama var. Si el valor de var es -1, debe aparecer en pantalla el texto ”uno negativo”; si el valor es 0, queremos que aparezca en pantalla el texto ”cero”; Si el valor de var es 1, debe aparecer en pantalla el texto ”uno positivo”. Finalmente, si el valor de var es cualquier otro, debe aparecer el texto ”otro valor”.

>>switch var
case -1
disp('uno negativo')
case 0
disp('cero')
case 1
disp('uno positivo')

otherwise
disp('otro valor')
end
Si el valor de var es 2, aparecerá en pantalla el texto "otro valor".

Capítulo 4

Cálculo Simbólico

4.1. Variables simbólicas

Las variables simbólicas se crean con la instrucción **syms**. Una vez declaradas las variables simbólicas, se pueden definir funciones simbólicas. Por ejemplo

>>syms x a

f=x*sin(x/a);

Nótese que en el cálculo simbólico se prescinde del punto que se anteponía con algunas operaciones para indicar que dicha operación se hacía coordenada a coordenada. Si en algún momento dudamos si una función es de datos o simbólica se puede recurrir a la instrucción **class**. Si tecleamos

>>class(x)

y pulsamos enter, obtenemos

ans sym

que nos advierte de que x es una variable simbólica.

4.2. Derivación e integración

1) CÁLCULO DE DERIVADAS. Para calcular derivadas, Matlab dispone de la función **diff**. Supongamos que se quiere calcular la derivada tercera de

$f(x) = x^3 \operatorname{sen}(x/a)$. Se procede como sigue:

>>syms x a

f=x^3*sin(x/a);

d3f=diff(f,3)

En pantalla aparece la derivada tercera de f escrita a la manera de Matlab. Si se quiere que aparezca expresada de la forma usual, se puede usar la instrucción **pretty**:

>>syms x a

f=x^3*sin(x/a);

d3f=diff(f,3)

pretty(d3f)

Para calcular la derivada primera de f basta escribir diff(f).

2) CÁLCULO DE PRIMITIVAS. Para calcular una primitiva de una función f, se dispone de la función **int**.

Ejemplo. Calcular $\int xsenx\, dx$.

>>syms x

f=x*sin(x);

F=int(f)

3) CÁLCULO DE INTEGRALES DEFINIDAS.

Ejemplo. Calcular $\displaystyle\int_0^2 xe^x\, dx$.

>>syms x

f=x*exp(x);

I=int(f,x,0,2)

ans exp(2)+1

Si se desea obtener el resultado en forma decimal y con un número preciso de decimales, se usa la instrucción **vpa(I,n)**, donde n es el número total de dígitos para representar el valor de I. Si ponemos

>>vpa(I,5)

ans 8.3891

En realidad, no es necesario indicar la variable independiente x y basta poner I = int(f,0,2).

4.3. Cálculo de límites

Para el cálculo de límites, Matlab dispone de la función **limit**.

Ejemplo. Calcular $\lim_{h\to 0} \dfrac{\operatorname{sen}(x+h)-\operatorname{sen} x}{h}$.

>>syms x h

f=(sin(x+h)-sin(x))/h;

a=limit(f,h,0)

ans $\cos(x)$

Ejemplo. Calcular $\lim_{x\to 0^+} \dfrac{sen\sqrt{x}}{\sqrt{x}}$.

>>syms x

f=sin(sqrt(x))/sqrt(x);

limit(f,x,0,'right')

Ejemplo. Calcular $\lim_{x\to\infty} \dfrac{x}{e^{x^2}}$.

>>syms x

f=x/exp(x^2);

limit(f,x,inf)

4.4. El polinomio de Taylor

Para determinar el polinomio de Taylor de una función f, de orden n en el punto $x = a$, Matlab dispone de la función **taylor**, cuya sintaxis es: taylor(f,n+1,a). Nótese que, si desea el polinomio de orden n, en la función taylor se indica $n + 1$, que es el número de términos de que consta dicho polinomio.

Ejemplo. Determinar el polinomio de Taylor de $f(x) = (x+1)\operatorname{sen}(x^2)$ de orden 3 en el origen

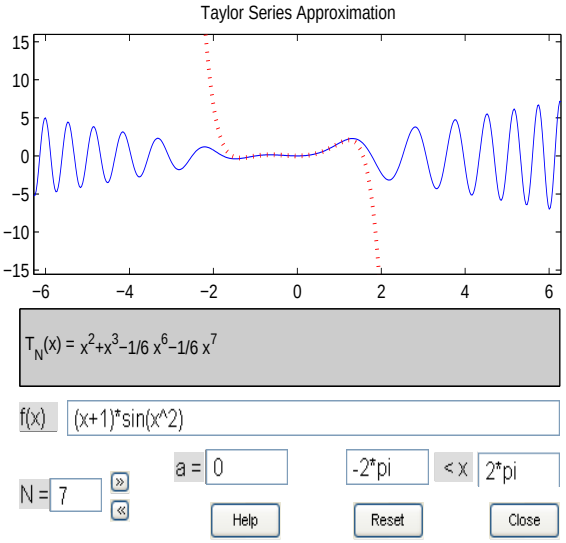
>>syms x

f=(x+1)*sin(x^2);

taylor(f,4,0)

ans $x^2 + x^3$.

Matlab dispone de otra función para obtener el polinomio de taylor. Se trata de la función **taylortool** que nos abre una ventana gráfica donde aparecen la gráfica de la función y del polinomio de taylor. Además, permite ir cambiando el orden del polinomio, el punto $x = a$, etc, manipulando adecuadamente los correspondientes "botones".



La gráfica de la función aparece en trazo continuo de color azul. Pulsando en el botón N se puede ir cambiando el orden del polinomio.

4.5. Resolución de ecuaciones algebraicas

Para encontrar las soluciones de una ecuación o (sistema) algebraica, Matlab dispone de la función **solve**.

Ejemplo. Resolver la ecuación $x^3 + x + 1 = 0$.

>>solve('xˆ3+x+1=0')

vpa(ans,4)

-0.6826

0.3411-1.162 i

0.3411+1.162 i

Ejemplo. Resolver el sistema $x + 2y = 1, 2x - 3y = 2$.

>>[x,y]=solve('x+2*y=1','2*x-3*y=2')

ans x=1, y=0.

También puede usarse solve para despejar una variable en una igualdad.

Ejemplo. Despejar b en la igualdad $3ab + c = 1$.

>>solve('3*a*b+c=1','b')

ans b=(1-c)/3a.

4.6. Raíces de ecuaciones arbitrarias

Para ecuaciones generales, Matlab dispone de la función **fzero**. Puede usarse de dos formas diferentes.

a) Si somos capaces de determinar un intervalo [a,b] que contenga la raíz que buscamos.

Ejemplo. Encontrar en el intervalo $[\pi/6, \pi]$ una raíz de la ecuación $\operatorname{sen} x - x^2 = 0$.

Sea $f(x) = \operatorname{sen} x - x^2$. Como $f(\pi/6) = \pi^2/36 - 1/2 = -0.2258 < 0$ y $f(\pi) = \pi^2 > 0$, el Teorema de Bolzano nos asegura que hay una raíz de la ecuación en el intervalo en cuestión. Para encontrarla, procedemos como sigue

>>fzero('sin(x)-x^2',[pi/6,pi])

ans 0.8767.

b) Partiendo de un valor cercano. La función fzero también puede encontrar una raíz de una ecuación a partir de un valor cercano.

Ejemplo. Determinar una raíz de la ecuación anterior con el dato $x_0 = \pi$.

>>fzero('sin(x)-x^2',pi)

ans 0.8767.

4.7. Resolución de ecuaciones diferenciales

Vamos a ver cómo se emplea la función **dsolve** para resolver ecuaciones diferenciales.

a) La integral general de una ecuación diferencial. Supongamos que se desea resolver la ecuación diferencial

$$y' = 1 + y^2.$$

>>y=dsolve('Dy=1+y^2','x')

ans $y = \mathrm{tg}(x + c_1)$.

Si no se indica la variable independiente 'x', Matlab nos da la respuesta con la variable t: $y = \mathrm{tg}(t + c_1)$.

b) Una integral particular. Supongamos que buscamos la solución de la ecuación

$$y''' - 3y'' + 3y' - y = 0$$

que verifica las condiciones iniciales $y''(0) = 1, y'(0) = 0$, e $y(0) = 0$.

>>y=dsolve('D3y-3*D2y+3*Dy-y=0','D2y(0)=1','Dy(0)=0','y(0)=0','x')

ans $y = 1/2x^2e^x$.

4.8. Sustitución y evaluación de expresiones

Empezaremos estudiando cómo se sustituyen en una expresión simbólica unas variables por otras expresiones simbólicas.

Ejemplo. Sustituir en $(x-2)^2 + y^2 - 1$ x e y por $\cos(t)$ y $\mathrm{sen}(t)$, respectivamente.

>>syms x y t

f=(x-2)^2+y^2-1;

g=subs(f,{x,y},{cos(t),sin(t)});

g

ans $(\cos(t) - 2)^2 + \mathrm{sen}^2(t) - 1$

Si queremos obtener una expresión simplificada de g

simple(g)

ans -4*cos(t) +4.

4.9. La Transformada de Laplace

Para encontrar la transformada de Laplace se usa la función **laplace**. Si no se indica otra cosa, expresa la transformada en función de la variable s.

Ejemplo. Encontrar la transformada de $f(y) = y^5$.

>>syms y

laplace(yˆ5)

ans $120/s^6$

Si la función tiene por variable s, entonces Matlab nos da la transformada con variable dependiente t.

Ejemplo. Encontrar la transformada de Laplace de $f(s) = \text{sen}(as)$.

>>syms a s

laplace(sin(a*s))

ans $a/(t^2 + a^2)$.

Si se desea que la variable dependiente de la transformada sea la que nosotros determinemos, por ejemplo p, se procede como sigue.

Ejemplo. Encontrar la transformada de $f(x) = \cos(ax)$.

>>syms a x p

laplace(cos(a*x),p)

ans $p/(p^2 + a^2)$.

Función delta de Dirac:

>>syms t

laplace(dirac(t-2))

ans e^{-2s}.

Función escalón unidad de Heaviside:

>>syms t

laplace(heaviside(t-3))

ans e^{-3s}/s.

Transformada de la derivada:

>>syms t

laplace(diff(sym('F(t)')))

ans s*laplace(F(t),t,s)-F(0).

4.10. La transformada inversa

La transformada inversa se obtiene con la función **ilaplace**. Por defecto, expresa la función con variable dependiente t.

Ejemplo. Encontrar la transformada inversa de $F(p) = 6/p^4$.

>>syms p

ilaplace(6/p^4)

ans t^3.

Se puede conseguir que la variable dependiente de la función sea la que queramos, por ejemplo, x.

Ejemplo. Transformada inversa de $F(p) = 1/(p-1)$.

>>syms p x

ilaplace(1/(p-1),p,x)

ans e^x.