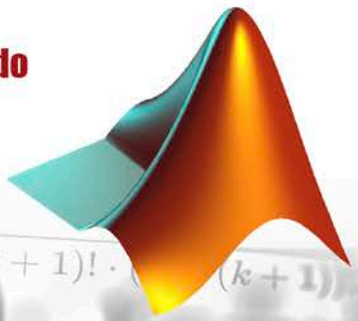
académica

Funciones de varias variables y ecuaciones diferenciales

Ejercicios resueltos analíticamente y con Matlab

Lucía Agud Albesa | Leonor Pla Ferrando
Macarena Boix García



Editorial
Universitat Politècnica de València

Lucía Agud Albesa
Leonor Pla Ferrando
Macarena Boix García

Funciones de varias variables y ecuaciones diferenciales. Ejercicios resueltos analíticamente y con Matlab

Editorial
Universitat Politècnica
de València

Colección *Académica*

Para referenciar esta publicación utilice la siguiente cita: Agud Albesa, L.; Pla Ferrando, L.; Boix García, M. (2020). *Funciones de varias variables y ecuaciones diferenciales. Ejercicios resueltos analíticamente y con Matlab.* Valencia: Editorial Universitat Politècnica de València

Venta: www.lalibreria.upv.es / Ref.: 0361_04_01_01

Imprime: Byprint Percom, S. L.

ISBN: 978-84-9048-945-1
Impreso bajo demanda

Si el lector detecta algún error en el libro o bien quiere contactar con los autores, puede enviar un correo a edicion@editorial.upv.es

Impreso en España

Resumen

Este libro presenta una recopilación de ejercicios resueltos dentro del marco de asignaturas de ciencias o ingenierías donde se vean funciones de varias variables y ecuaciones diferenciales.

Los contenidos tratados serán:

1. Introducción a las curvas en el plano y superficies en el espacio. Dominios en el plano. Cambios de coordenadas: polares, cilíndricas y esféricas. Derivación de funciones de varias variables.

2. Cálculo de volúmenes y áreas de superficies.

3. Ecuaciones diferenciales de orden 1. Métodos numéricos para la resolución de EDOs de orden 1.

4. Ecuaciones diferenciales de orden 2 con coeficientes constantes. Transformada de Laplace.

La estructura general será plantear un ejercicio del cual se encontrarán dos resoluciones: una analítica con todos los pasos explicados y otra realizada mediante el paquete matemático `Matlab` versión `R2019b` y `R2020a`. Cuando se termina la resolución de un ejercicio de forma analítica el lector se encontrará con el símbolo □, mientras que cuando se termina la resolución de un ejercicio usando `Matlab` se indicará con el símbolo ⊠.

Todas las instrucciones que se van a ejecutar con `Matlab` pueden encontrarse explicadas con más detalle en (Agud Albesa y Pla Ferrando 2015), aunque a lo

largo de este libro se irán explicando las órdenes que se trabajen y comentarios sobre las mismas.

La resolución mediante `Matlab` de ejercicios similares se hará sólo de uno de ellos, ya que los pasos e instrucciones serán los mismos cambiando los argumentos de entrada. Se indicará el ejercicio de referencia donde esto ocurra.

Una de las principales motivaciones que ha llevado a las autoras a escribir este libro es ayudar al alumno a comprender, y no mecanizar, la resolución de problemas matemáticos, cómo afrontarlos tanto de forma analítica como con la ayuda de un paquete matemático como es en este caso `Matlab`. También ese contacto que todo estudiante o persona que se acerca a las ciencias debe mantener con el lenguaje científico, que es universal y que permite, independientemente de nacionalidades, que toda la comunidad científica pueda expresarse y entenderse de forma única y unívoca.

Por otro lado, la sociedad actual es una sociedad *online*, que pone al alcance de los ciudadanos cada vez más herramientas para realizar muchos de los pasos y cuentas que antes había que hacer 'a mano'. Hay que potenciar las ventajas que esto ofrece, pero dándose cuenta de que es el usuario el que le dice a las máquinas, a las herramientas, a los paquetes matemáticos, qué es lo que se debe hacer. Si no hay una comprensión de porqué se realizan las operaciones que se muestran, o qué operaciones matemáticas son las que permiten llegar al resultado adecuado, de nada sirve un paquete matemático que ejecute esas instrucciones. Es por eso que las autoras han visto interesante completar la resolución de los ejercicios mediante `Matlab`, indicando qué debe hacerse y con qué órdenes `Matlab` permite llegar al resultado deseado.

En las ocasiones que interese añadir resultados teóricos o propiedades, se pondrán en la sección o tema correspondiente.

El capítulo correspondiente a métodos numéricos para ecuaciones diferenciales de orden 1 será desarrollado de forma teórica, ilustrando cada uno de los métodos explicados mediante ejercicios resueltos.

Algunos de los enunciados de los ejercicios de EDOs aquí expuestos se han obtenido de (Bellido Guerrero, Donoso Bellón y Lajara López 2014) y (Ricardo 2008).

Así mismo, el tema de ecuaciones diferenciales de orden 2 el lector lo encontrará desarrollado teóricamente de forma más exhaustiva, pensando en que cualquier persona interesada en el tema o que cualquier alumno de cualquier titulación de ciencias que se acerque a este libro pueda entender de dónde proceden

los mecanismos que se ejecutan en la resolución de las EDOs de orden 2 con coeficientes constantes.

El libro termina con un capítulo donde se han incluido los comandos de las diversas figuras de representaciones gráficas con las que se han ilustrado los ejercicios, de cara a que aquel lector que quiera profundizar en el uso del `Matlab` tenga aquí cómo llevar a cabo esas representaciones. Así de paso, se van introduciendo más comandos de `Matlab` y cómo trabajarlos. Indicar que los comandos necesarios para las representaciones de las superficies o de los dominios de integración, se incluyen en la resolución de `Matlab` del propio ejercicio. Este último capítulo se centra en instrucciones que completan y complementan estas representaciones gráficas.

Esperamos que el lector pueda encontrar en este libro una ayuda para comprender mucho mejor los conceptos matemáticos aquí tratados y poder llevar a cabo su resolución, tanto analíticamente como con la ayuda de `Matlab`, así como saber expresar y entender en el lenguaje científico cualquier tipo de problema de los campos aquí tratados.

Índice general

Capítulo 1

Introducción a Matlab

1.1 Entorno Matlab

Este libro utiliza la versión de Matlab R2019b y R2020a. La ventana que el usuario se encontrará al abrir este paquete matemático se subdivide en varias ventanas dedicadas a mostrar distintas partes de la sesión de trabajo, Figura 1.1, aunque la configuración inicial de estas puede cambiar. Se recomienda al usuario de Matlab tener abiertas, además evidentemente de la *Command Window* o ventana de trabajo principal donde se introducen las instrucciones y se ven los resultados, las siguientes ventanas:

- *Current Folder*
- *Workspace*
- *Command History*

y colocarlas como se indica en la Figura 1.1, sobre todo para ver enseguida la ventana que muestra las variables que se van definiendo y el historial de las instrucciones que se van ejecutando. Estas ventanas se describen con más detalle a continuación.

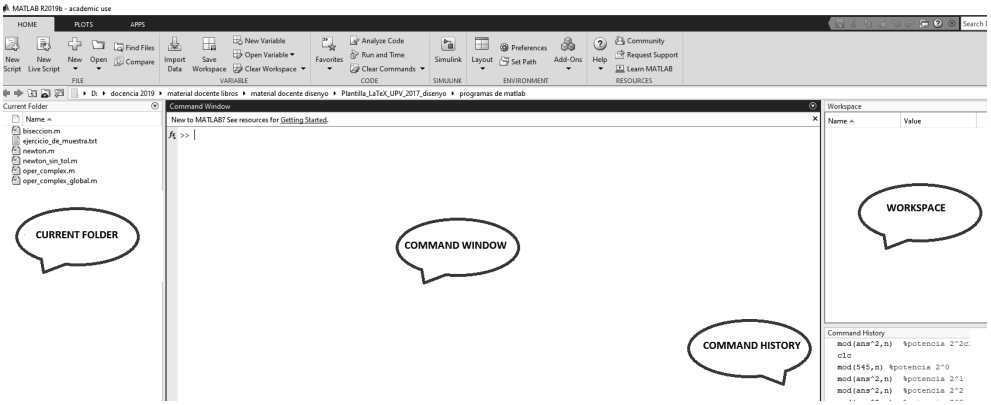


Figura 1.1: Ventana de trabajo del paquete matemático `Matlab`

La *Command Window* (ventana central) es donde se introducen los comandos, variables e instrucciones a realizar. Es decir, es la ventana donde se trabaja. Ahí `Matlab` devuelve los resultados de cualquier instrucción ejecutada o de cualquier programa que se haya hecho funcionar.

La ventana *Current Folder* (superior izquierda) indica el contenido del directorio en el que se está trabajando en esa sesión y, salvo cambio del mismo, es donde se irán guardando los archivos que se vayan salvando. Si se quiere acceder a algún fichero que ya se tiene creado, debe tenerse en cuenta que para ejecutarlo desde la ventana de *Command Window* y que `Matlab` lo encuentre, siempre buscará en la ruta de acceso que se muestra en la barra del *Browser*. Por eso debe prestarse atención a estar en la ruta de trabajo adecuada, de no ser así `Matlab` indicará que tal fichero no ha sido encontrado en esa dirección.

La ventana de *Workspace* (superior derecha) es la ventana donde se indican las variables que se van definiendo en la sesión de trabajo o que se tienen guardadas y cargadas de otras sesiones. Para borrar alguna de ellas, hay que usar el comando >> `clear` seguido del nombre de la variable o seleccionarla en esa ventana y eliminarla. Esta ventana es importante, ya que cuando el lector vaya ejecutando los distintos ejemplos, quizás ya tenga definida la variable que precisa para las instrucción a ejecutar y así no hace falta volver a definirla. Es conveniente siempre consultarlo en esta ventana por si ya tuviera un valor adjudicado de alguna asignación anterior.

La ventana de *Command History* muestra todos los comandos y órdenes introducidos, permitiendo recuperarlos, o bien arrastrándolos a la *Command Window* (no ejecuta), o bien haciendo doble click sobre ellos (ejecuta).

1.1.1 Figuras

Cuando se ejecuta el comando para representar una función, `Matlab` abrirá una nueva ventana de *Figure*, que se puede minimizar y mantener toda la sesión mientras se va actualizando, o bien cerrar. Cuando se representa otra figura reemplaza la anterior, salvo que se abra una nueva ventana ejecutando el comando

```
>>figure
```

aunque también se le puede indicar que represente en la misma ventana la siguiente gráfica, usando el comando

```
>>hold on
```

Si se quiere desactivar este comando basta introducir >> `hold off`.

Para saber un poco más sobre la representación gráfica, además de todas las indicaciones que se han ido añadiendo en los ejercicios resueltos con `Matlab`, el lector puede acudir al Capítulo 6, donde ya se habla más extensamente de cómo representar y de cómo manipular las gráficas.

1.1.2 Archivos: creación y cómo guardarlos

Con el icono del folio o *New Script* de la barra de herramientas (primero de los iconos de la misma), también se abrirá una nueva ventana donde `Matlab` permite programar o crear funciones en ficheros con extensión *.m*. Estos archivos se guardan desde esa ventana de *Editor* donde el usuario encontrará la opción de *Save* o el icono correspondiente (generalmente el disquete).

A lo largo del libro, y de otros libros de las autoras (Agud Albesa y Pla Ferrando 2020), el lector encontrará algún ejemplo de ficheros .m creados por las mismos para que vea cómo manipularlos.

Es importante el nombre con el que se guarde este tipo de ficheros ya que cuando se vaya a ejecutar desde la *Command Window*, o dándole a *Run* en la ventana del *Editor*, es con ese nombre con el que se les llama. Por lo tanto, si el arhivo es una *function*, un programa donde hay argumentos de entrada

que se piden con el nombre, el nombre de la *function* y del fichero debe ser el mismo. Por defecto `Matlab` ya les adjudica esa primera línea del código.

A veces `Matlab` permite acceder al código de programación de funciones internas suyas, lo que puede ayudar al usuario que se acerca por primera vez a la programación con `Matlab`. Se puede hacer o bien con la instrucción

```
>>edit Nombre_funcion
```

con lo que abrirá esa función en la ventana del *Editor*, Figura 1.2, o bien pidiendo

```
>>type Nombre_funcion
```

donde aparecerá el código de la función en la propia ventana de la *Command Window*. De la primera forma queda más claro, ya que `Matlab` usa colores para los distintos tipos de sentencia (verde para comentarios, -introducidos con el %-, morado para estructuras de programación, negro para instrucciones básicas, morado para opciones de los argumentos, etc.)

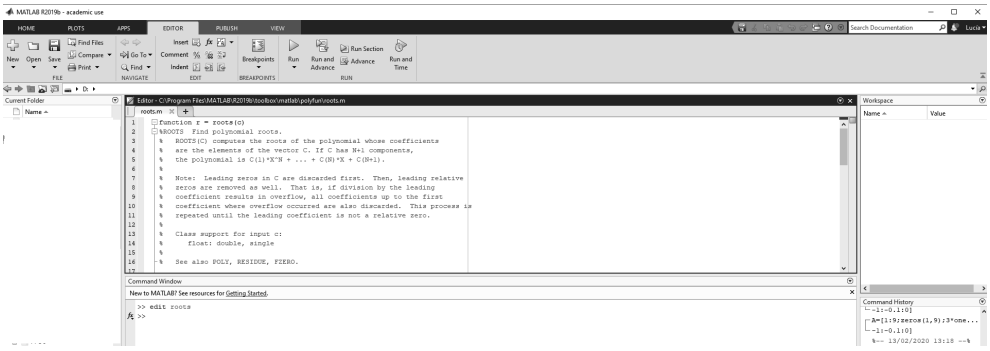


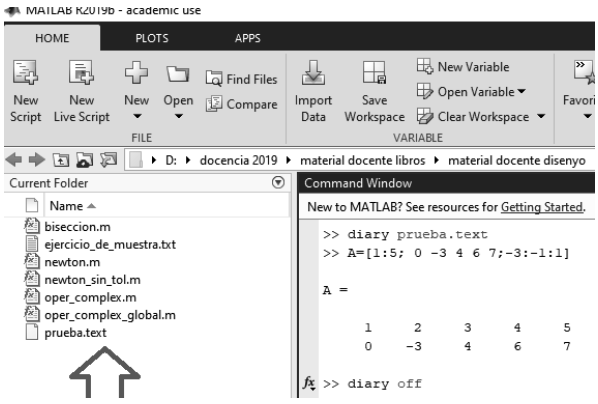
Figura 1.2: Acceso a una *function* propia de `Matlab` en el *Editor*

Evidentemente no siempre permite el acceso al código de sus programas.

La sesión de trabajo de la *Command Window* se puede guardar. Para ello, desde el instante en que se quiera guardar, escribir (sin espacios en blanco en el nombre del fichero):

```
>>diary nombre_fichero.txt
```

admitiendo también extensión .text. Esto queda indicado en la figura siguiente, donde se ha dado nombre a una sesión y puede verse en la ventana de *Current Folder* cómo automáticamente queda creado el archivo de texto con ese nombre



Desde ahí, hasta indicar >> `diary off`, guarda con el nombre indicado la sesión realizada. Si en cualquier momento se quiere volver a activar dicha sesión, basta con poner de nuevo >> `diary on`. Por defecto se guarda en la ruta especificada en la barra de *Browser*. Este fichero es un documento de texto, que se puede abrir desde el *Bloc de Notas*, o haciendo doble click desde la sesión de `Matlab` en el directorio donde está. En este último caso abre el documento en la ventana de *Editor* de `Matlab`, ver Figura 1.3.

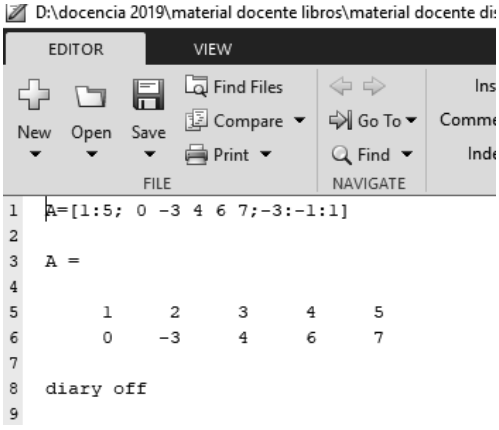


Figura 1.3: Editor con el contenido de la sesión guardada al pulsar dos veces en el archivo

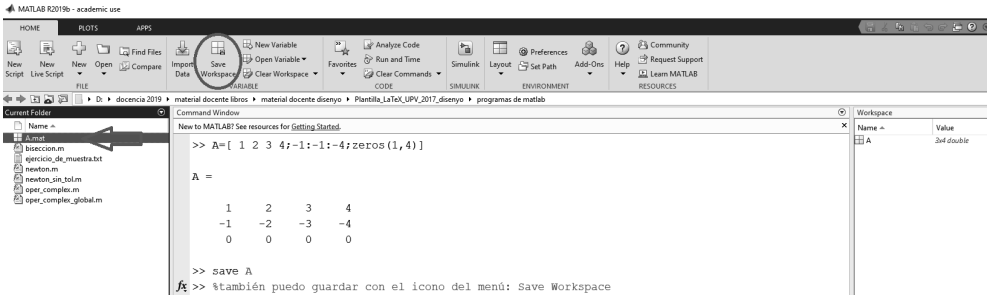
Las instrucciones ejecutadas después de la orden >> `diary off` no se guardan en el documento correspondiente a esa sesión.

Es importante indicar que al guardar una sesión en la que se hayan representado gráficas, estas no se van a guardar con la instrucción `diary`, ya que como el usuario puede observar, las gráficas `Matlab` las representa en una ventana aparte. Con lo cual, salvo la instrucción para ejecutarlas, no quedan archivadas como sesión de trabajo. Si se quieren guardar, desde la propia ventana de la figura representada deberán salvarse, o copiarse en otro documento, ofreciendo numerosas extensiones: .fig (extensión por defecto de `Matlab` que cuando se vuelva a abrir desde la ventana de trabajo permitirá volver a manipularlas), .jpg, .png, .eps, y muchas otras. De esto se habla de forma más extensa en el Capítulo 6.

Al final de ese capítulo se presenta otra forma de trabajar en `Matlab` pudiendo solventar el hecho de que no se guarden las gráficas, que sí se guarden los mensajes de error en la sesión de trabajo de la *Command Window* y se verá una forma de intercambiar fácilmente escritura de texto y de comandos en un mismo archivo. Basta trabajar con *Live Script*, que abre un *Live Editor* y permite realizar todo esto en un único archivo con extensión .mlx, que además se puede exportar como pdf. Se hablará con detalle de esto en la Sección 6.4.

1.1.3 Cómo guardar las variables

Si se quiere guardar las variables creadas en la sesión -después se habla un poco más sobre las variables en `Matlab`-, bastará con el comando o instrucción >> `save nombre_variable`; o bien dentro del menú de *File* seleccionar *Save Workspace*, o incluso desde el icono con este mismo nombre disponible en la barra de herramientas (rodeado en la figura siguiente). Los archivos de variables tienen por defecto extensión *.mat* y al guardarlos deben aparecer en la ventana de *Current Folder*.



Para recuperar las variables en una nueva sesión de trabajo deben cargarse con el comando >> `load` y el nombre del fichero de variables que se haya especificado al guardarlas. Si se quiere, además, ver estas variables, haciendo doble click en la variable elegida de la ventana de *Workspace*, `Matlab` abre una nueva ventana tipo hoja de cálculo donde están almacenadas las variables como se observa en la figura siguiente

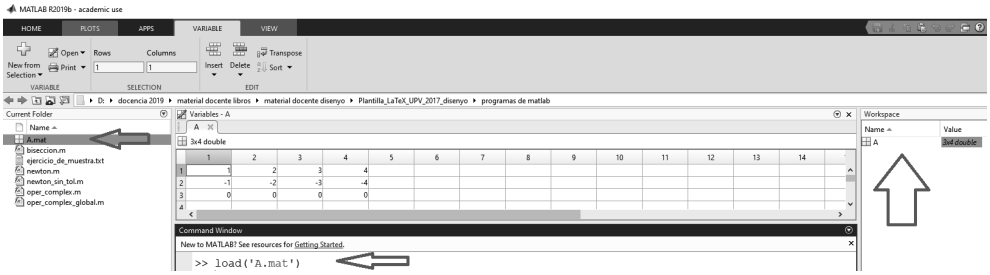


Figura 1.4: Ventana dentro de `Matlab` donde se almacenan las variables. Se marcan en la figura las variables seleccionadas para ser mostradas y cómo cargarlas

1.2 Comandos o instrucciones en `Matlab`

Los comandos en `Matlab` siempre estarán escritos en minúscula y entre paréntesis sus argumentos. Su escritura es en inglés y se ejecutan al dar a la tecla *Enter*. Si se tienen dudas de los argumentos de alguna instrucción, bastará con escribir en la ventana de comandos:

```
>>help nombre_comando
```

y saldrá la ayuda de `Matlab` . Por ejemplo, con >> `help gcd`, `Matlab` dará la ayuda que tiene para el cálculo del máximo común divisor. Para saber más sobre la ayuda que proporciona `Matlab` puede verse la Sección 1.4.

Las instrucciones pueden ir cada una en una línea y ejecutarse por separado, o en la misma línea separadas por comas (se produce además salida por pantalla) o por punto y coma (no se ve la salida por pantalla).

Para recuperar alguno de los comandos introducidos, hay dos formas. Una, como ya se ha comentado, desde la ventana de *Command History*; la otra, con las teclas de desplazamiento. Las flechas de arriba y abajo, $\uparrow$, $\downarrow$ recuperan los comandos. Mientras que las de izquierda y derecha, $\leftarrow$, $\rightarrow$ mueven el cursor dentro de la línea de edición para poder modificar las instrucciones.

Para interrumpir el funcionamiento de una instrucción de `Matlab` se pulsan las teclas *Control* y *C* a la vez. A veces, esta operación se deberá repetir.

Para salir de `Matlab` o bien cerrar la ventana lo podemos hacer bien con los comando >> `exit`, o >> `quit`, dándole al *Enter* al final de cualquier instrucción para que así se ejecute.

Si al introducir una orden en una línea de la *Command Window* no cupiese toda entera se puede terminar con puntos suspensivos, (...), y darle a *Enter*. `Matlab` entiende que no ha terminado la instrucción y sin aparecer el *prompt* del sistema, >>, deja seguir escribiendo en la línea siguiente, ejecutando la instrucción al pulsar *Enter*. Por ejemplo

```
>>A=[1:11;zeros(1,11);3*ones(1,11);-1:0.1:0]
A =
>> A
A =
 1     2       3      4       5       6       7       8        9        10         11
 0     0       0      0       0       0       0       0        0        0          0
 3     3       3      3       3       3       3       3        3        3          3
-1    -9/10   -4/5   -7/10   -3/5    -1/2    -2/5    -3/10    -1/5      -1/10      0
```

Si en una línea de instrucción se pone el símbolo %, cambia el color del texto a verde y automáticamente `Matlab` entiende que lo que se escriba hasta darle al *Enter*, es un comentario y no debe ser ejecutado.

Interesante saber que, si una vez ejecutadas muchas instrucciones, se quiere limpiar la pantalla pero no borrar las variables, el comando a utilizar es

```
>>clc
```

1.3 Variables y formatos

Cuando se ejecuta un comando y no se le da nombre a la variable que se obtiene como resultado, `Matlab` lo asigna a una variable que él tiene en el sistema llamada `ans`. Como este nombre ya lo tiene asignado, nunca se puede llamar a ninguna variable con este nombre. En cada ejecución sin variable de salida, `Matlab` irá guardando en ella ese resultado machacando el resultado anterior. Si se quiere guardar en una variable la operación ejecutada, se asigna con el nombre deseado y el símbolo = (asignación, no comparación para lo que se escribiría ==)

```
>>a=gcd(3,12)
a=
3
```

Automáticamente en la ventana del *Workspace* aparecerá la variable `a` e indicará su tipo y dimensión. Para asignar a esta variable otro valor, simplemente o se borra o se reasigna con otro valor. Cabe recordar que `Matlab` es sensible a mayúsculas y minúsculas; por lo tanto a y A son dos variables distintas.

No se puede nombrar a una variable cuyo nombre ya esté siendo utilizado por `Matlab` o bien en una función o bien para sus variables internas. Variables ya asignadas por `Matlab` son:

- `i` ó `j` ... para la unidad imaginaria $i = \sqrt{-1}$
- `pi` ... para el valor de π
- `ans` ... para las variables de salida que no tengan asignación previa
- `eps` ... su valor es $2{,}2204e-016$. También puede utilizarse como comando (mirar >> `help eps` en caso de querer más información)

Para borrar una variable hay varias formas. Basta nombrar a otra variable con ese nombre y se reemplaza su valor por el nuevo. También puede seleccionarse la variable en la ventana de *Workspace* y darle a *Delete* con las opciones del botón derecho del ratón, o con la tecla de suprimir del teclado. Pueden seleccionarse varias variables. También existe el comando

```
>>clear nombre_variable
```

Si solo se indica >> `clear`, `Matlab` borra de memoria todas las variables definidas hasta el momento.

1.3.1 *Almacenamiento interno de variables en* `Matlab`

`Matlab` usa para el almacenamiento interno de los números, coma flotante normalizada; es decir, notación científica de forma que la parte entera es 0 y la primera cifra decimal es distinta de cero (0.00003 sería 0.3*e-04).

Generalmente para las salidas por pantalla trabaja con el formato *short*, 4 dígitos decimales

```
>>0.00003
ans =
3.0000e-05
```

no se puede confundir lo que `Matlab` saca por pantalla con su manejo interno de los datos. De ahí que cuando se quiera precisión con números irracionales conviene definirlos como simbólicos.

1.3.2 Cómo modificar las salidas por pantalla

Si se quiere cambiar el formato de una salida por pantalla, no el interno de almacenamiento de `Matlab`, se pide con el comando

```
>>format tipo_de_formato
```

Los formatos más usuales en `Matlab` son:

- FORMAT SHORT: 5 dígitos, contando parte entera y decimal
- FORMAT LONG: 15 dígitos
- FORMAT SHORTE: 5 dígitos en coma flotante normalizada
- FORMAT LONGE: 15 dígitos o 7 en coma flotante normalizada
- FORMAT SHORTG: elige el mejor formato con 5 dígitos de salida (adecuado si se trabaja en un vector con números de diferentes longitudes)
- FORMAT +: saca por pantalla los signos, +, - y espacios en blanco
- FORMAT RAT: pone el valor en forma racional

Por ejemplo, el número π con varios formatos sería

```
>>pi
ans =
3.1416
>>format shorte,pi
ans =
3.1416e+00
>>format long,pi
ans =
3.141592653589793
```

Si se quiere saber más sobre formatos de `Matlab` basta escribir

```
>>help format
```

En las últimas versiones de `Matlab` se ha visto disminuida la cantidad de información que se obtiene con la ayuda.

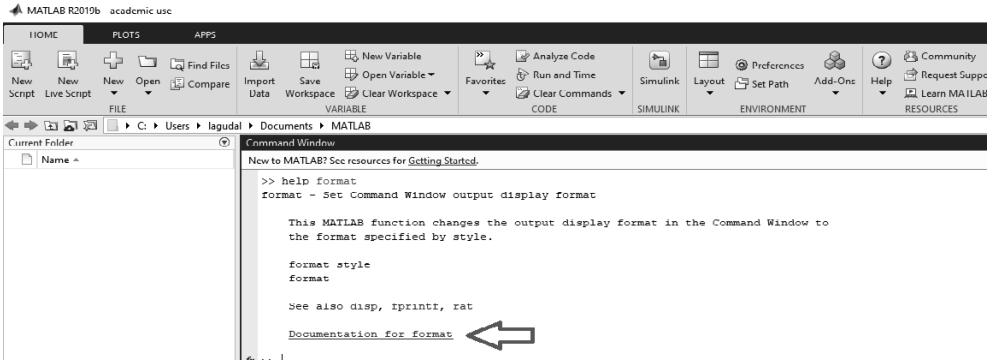


Figura 1.5: Salida por pantalla de la ayuda de `Matlab`

Para acceder a más ejemplos e información basta acudir al documento del enlace señalado en la Figura 1.5, obteniendo

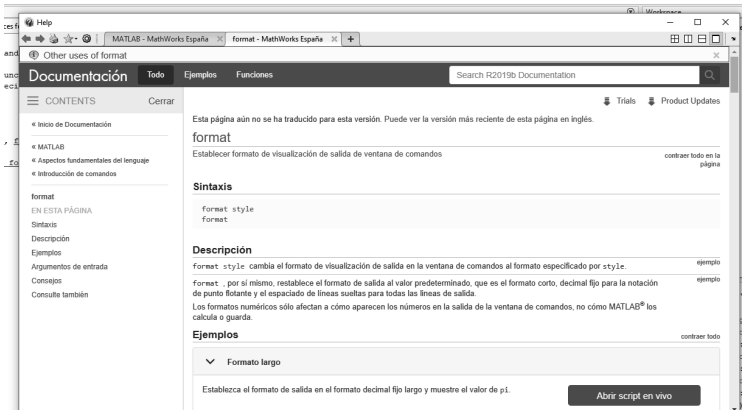
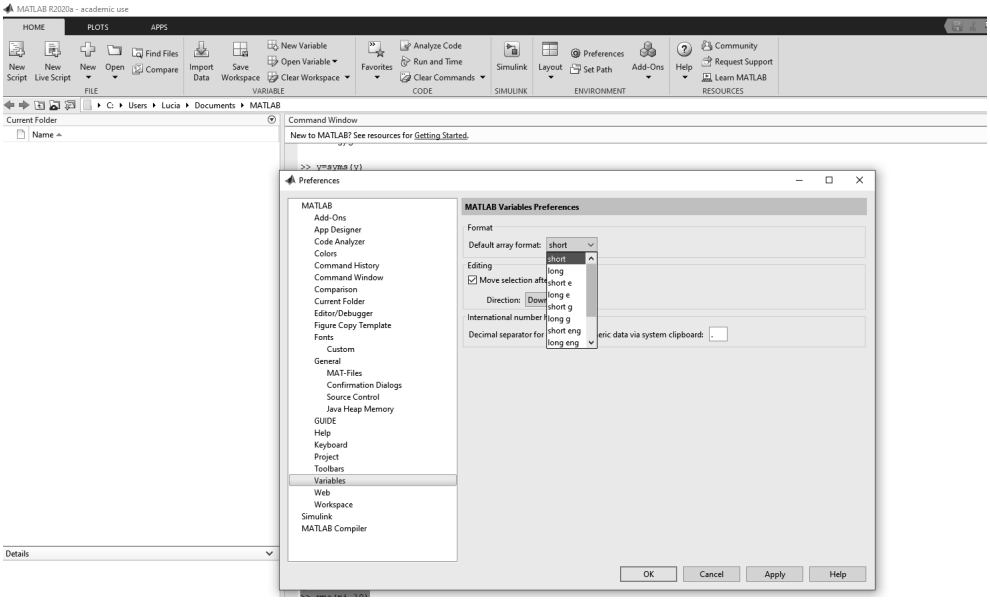


Figura 1.6: Salida por pantalla de la ayuda de `Matlab` para la instrucción >> `format`

Esta salida por pantalla se puede cambiar de forma definitiva para todo el paquete matemático `Matlab` desde *Preferences*, en los iconos de la barra de herramientas de `Matlab`



Y ahí aparece en *Format* un desplegable con los distintos formatos de salida por pantalla. Eso afectaría a todas las sesiones de trabajo posteriores hasta que se vuelva a cambiar.

Nota 1.3.1 *Cabe destacar la diferencia entre las órdenes para declarar el formato de una variable o para cambiarlo a esa variable concretamente. Por ejemplo, si un número que ha devuelto* `Matlab` *quiere verse en formato racional puede hacerse de dos formas*

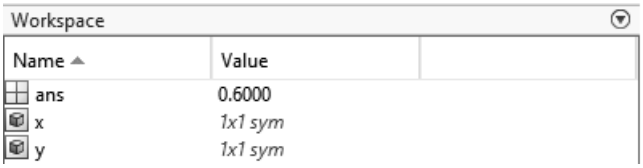
```
>>format short,3/5
ans =
0.6000
>>rats(ans) % convierte a racional ese valor únicamente
ans =
'       3/5      '
>>format rat,3/5 % cambia a racional todas las salidas posteriores
ans =
3/5
```

La orden `format` *cambia el formato para todas las salidas posteriores, hasta que se vuelva a indicar un nuevo formato. Con la instrucción aplicada a esa variable sólo actúa en ese momento, devolviéndola entre comillas simples. Obsérvese también que la orden se diferencia de la declaración de la variable en la letra final s:* `rats('variable')` *o* `format rat`.

Lo mismo sucede si se declara una variable como simbólica, para toda la sesión hasta que sea eliminada,

```
>>y=sym('y')
y =
y
>> syms x
```

En la ventana de Workspace ambas variables constan como simbólicas.



1.3.3 Prefijar el número de cifras

En caso de querer que un número salga por pantalla con una cantidad de cifras predeterminada existe la instrucción:

```
>>vpa(valor,numero_cifras)
```

donde `valor` es un dato numérico y el argumento `numero_cifras` indica cuántas cifras tendrá el número incluyendo la parte entera:

```
>>pi %salida por defecto
ans =
3.1416
>>vpa(pi) %si no se pone argumento
ans =
3.1415926535897932384626433832795
>>vpa(pi,10) % con 10 cifras
ans =
3.141592654
```

El resultado mostrado por pantalla será simbólico, puede comprobarse en la ventana de *Workspace*, Figura 1.7.

Si no se indica segundo argumento, la cantidad de cifras que `vpa` saca por defecto es 32 cifras, incluyendo la parte entera si no es nula, y en caso de ser nula, serían 32 cifras decimales. Este cantidad de cifras puede modificarse, siendo esto interesante cuando `vpa` se aplica a valores simbólicos, como por ejemplo es el caso detallado en la Figura 1.7. Para ello se utiliza el comando

```
>>digits(n)
```

siendo n la cantidad de dígitos que quieren obtenerse en la salida por pantalla por defecto, es decir, cuando no se especifica segundo argumento en `vpa`.

```
>>vpa(1/3) %32 dígitos
ans =
0.33333333333333333333333333333333
>>digits(5);vpa(1/3) %saldrán 5 dígitos (no cuenta la parte entera nula)
ans =
0.33333
```

Si llegado un momento no se sabe cuántas cifras está usando la instrucción `vpa`, basta pedirle a `Matlab`

```
>>digits
Digits = 32
```

aunque cuando se cierra una sesión de `Matlab` vuelve a la cantidad original.

Cabe recordar, que si la salida de esta instrucción se almacena en una variable, por ejemplo, a, esta variable es automáticamente simbólica (puede comprobarse mirando la ventana de *Workspace*). Si después ese valor quiere pasarse a numérico, basta utilizar la instrucción

```
>>a=double(valor_salida_de_vpa)
```

convirtiendo así la variable simbólica en numérica, concretamente con doble precisión.

La instrucción `vpa` también puede ser aplicada a variables simbólicas; prestando atención a que esa salida seguirá siendo simbólica, como puede observarse en la ventana de *Workspace*, Figura 1.7 donde se han ejecutado las siguientes instrucciones:

```
>>syms x,f(x)=3*x^2+exp(x-1);valor=f(3)
valor =
exp(2) + 27
>>%ahora lo paso a numérico
>>double(valor)
ans =
34.389056098930652
>>% lo paso a otro formato
>>format long,double(valor)
ans =
34.389056098930652
>>valor_2=vpa(valor,6) %ver en el workspace que esta variable es simbólica
valor_2 =
34.3891
>>double(valor_2) %numérico de nuevo para poderla evaluar o sustituir
ans =
34.389056098930268
```

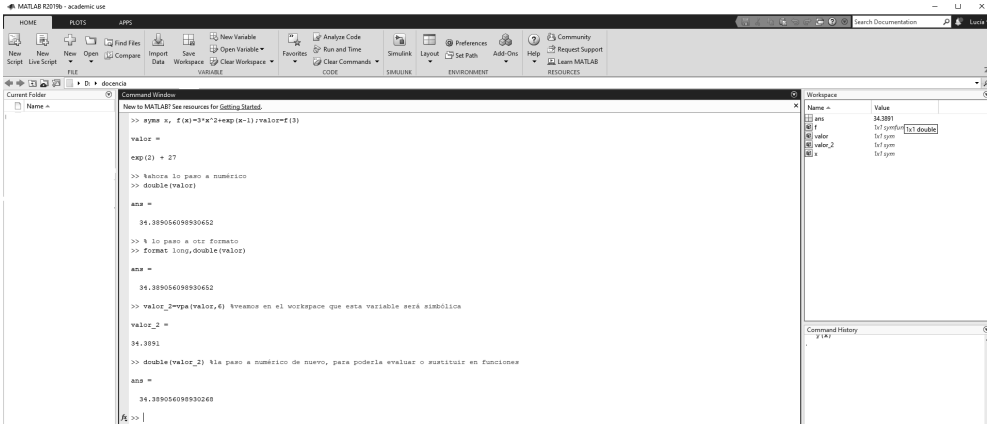


Figura 1.7: Comando >> `vpa` con variables simbólicas

Obsérvese que si ese valor obtenido se quiere sustituir o evaluar en alguna función para trabajarlo como numérico, debe transformarse utilizando de nuevo la instrucción >> `double(valor_2)`.

Si se busca que la salida por pantalla de una variable simbólica tenga sus coeficientes con formato decimal, la instrucción `vpa` actúa pero no hará caso al número de cifras que se le pida como segundo argumento; habrá que cambiar los dígitos por defecto de esta instrucción de forma previa

```
>>syms x,f(x)=1/3*x^2+3/7*x
 f(x) =
 x^2/3 + (3*x)/7
 >>f_decimal(x)=vpa(f)
 f_decimal(x) =
 0.333333333333333333333333333333333*x^2 +0.428571428571428571428571428571143*x
 >>f_decimal(x)=vpa(f,6) %no hace caso al segundo argumento
 f_decimal(x) =
 0.333333333333335701809119200333395*x^2 +0.428571428571430601550673600286625*x
 >>digits(5),f_decimal(x)=vpa(f) %cambia a salida de 5 cifras no nulas
 f_decimal(x) =
 0.33333*x^2 + 0.42857*x
```

Destacar también que esta instrucción no cuenta como cifras las nulas, es decir, si por ejemplo, se ejecuta

```
>>digits(5);vpa(1/30)
ans =
0.033333
```

se obtiene una salida por pantalla con 5 cifras no nulas. No con 5 cifras el número ni con 5 decimales.

1.4 Ayuda de `Matlab`

Hasta ahora se ha hablado del comando >> **`help nombre_de_comando`** que proporciona ayuda del comando solicitado.

Si lo que se quiere es obtener ayuda general de `Matlab`, o de las funciones o *Toolbox* (paquetes de herramientas) que `Matlab` tiene ya predefinidas, basta con darle al icono de *Help* que se señala en la siguiente figura, que abrirá la ventana de ayuda también mostrada en esta figura

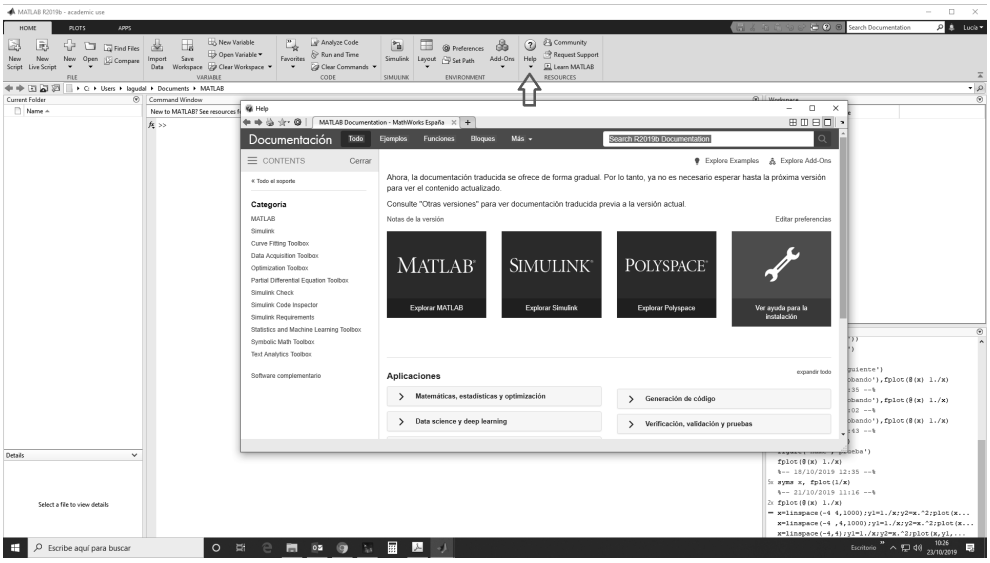
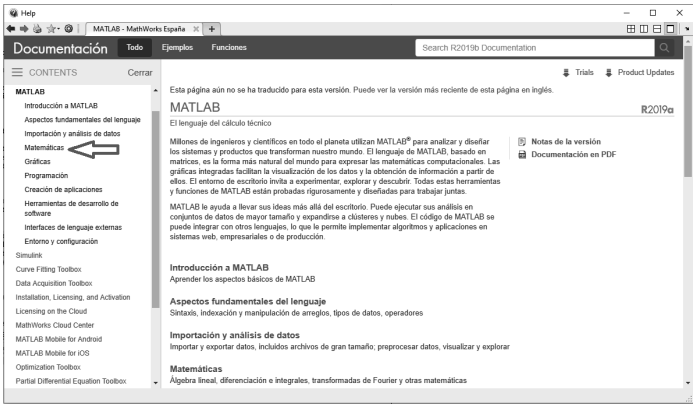


Figura 1.8: Ventana emergente de ayuda de `Matlab`

Si, por ejemplo, se quiere acceder a todas las funciones matemáticas que ya tiene construidas `Matlab`, bastaría darle al ítem dentro del listado de Categorías (a la izquierda de la ventana mostrada en la Figura 1.8) donde pone `Matlab` -destacado en la Figura 1.8-. Dentro de esta categoría, aparecerá en el índice Matemáticas, y pulsando ahí se accede a todas las funciones matemáticas y operaciones que `Matlab` realiza. Puede verse en la siguiente figura



Debe destacarse que, para poder acceder a toda esta información que `Matlab` proporciona, el usuario debe estar registrado en una cuenta de `Matlab`. Basta dar un *email* y una contraseña, siempre que se tenga licencia para poder descargarse este paquete, o bien personal, de empresa o de campus. La primera vez que se accede a la ayuda saldrá una pantalla en blanco, sólo con los márgenes, y al volver a pedir la ayuda pedirá que se registre. Una vez registrado, ya siempre puede acceder a la ayuda de `Matlab`. Esto ha cambiado con respecto a versiones anteriores.

Para más detalle e información sobre `Matlab` se recomienda acudir a los primeros capítulos del libro indicado en (Agud Albesa y Pla Ferrando 2015), aunque trabaja con versiones anteriores, hay algunas instrucciones que no se han visto modificadas. De todas formas, en cada una de las resoluciones de los ejercicios se va indicando al lector todo lo que debe ir sabiendo de `Matlab` para ejecutarlos correctamente.

1.5 Apéndice: soluciones numéricas de ecuaciones con `Matlab`

Si en alguna ocasión el lector se encuentra con ecuaciones que no puede resolver de forma analítica o simbólica mediante `Matlab` va a indicarse en esta sección alguna opción para solventar este problema.

Aunque en este libro de los métodos numéricos que se hablan son de aquellos que resuelven EDOs, se ha querido hacer aquí este inciso por si el lector a la hora de resolver ecuaciones que son necesarias para puntos de corte de dominios, etc. se encontrase con alguna ecuación no resoluble de forma analítica.

Entrar en este tema de manera más profunda sería hablar de Análisis Numérico y de métodos numéricos para ecuaciones, tema del cual se ha hablado en (Agud Albesa y Pla Ferrando 2020) explicando e implementando para `Matlab` diversos métodos numéricos por sus autoras. Pero si el lector sólo quiere conocer de qué herramientas dispone `Matlab` para solventar este problema, se hablará aquí de dos comandos que pueden ayudar a salir del paso.

1.5.1 Comando `vpasolve`

Para saber más sobre este comando basta acudir a la ayuda que `Matlab` proporciona

```
>>help vpasolve
--- help for vpasolve ---

vpasolve - Solve equations numerically

This MATLAB function numerically solves the equation eqn for the variable var.

S = vpasolve(eqn,var)
S = vpasolve(eqn,var,init_param)
Y = vpasolve(eqns,vars)
Y = vpasolve(eqns,vars,init_param)
[y1,...,yN] = vpasolve(eqns,vars)
[y1,...,yN] = vpasolve(eqns,vars,init_param)
___ = vpasolve(___,'Random',true)

See also dsolve, equationsToMatrix, fzero, linsolve, solve, symvar, vpa

Documentation for vpasolve
```

Actúa de distinta forma si la ecuación a resolver es polinómica o no. Por ejemplo,

```
>>syms x,vpasolve(x^4-x^3+2*x-1)
ans =
-1.153721375541767900865992748763 9
0.535687386791873052661405914398 23
0.809016994374947424102293417182 82 - 0.981593343275320473051353386619 8i
0.809016994374947424102293417182 82 + 0.981593343275320473051353386619 8i
```

devolviendo las 4 soluciones que debe tener un polinomio de grado 4. Obsérvese que devuelve 2 soluciones complejas (una y su conjugada) por ser un polinomio con coeficientes reales, y dos reales. Por defecto `Matlab` trabaja con una tolerancia de 10^{-16}, por lo que se pueden asegurar hasta 15 cifras decimales en la solución aproximada que devuelve un método numérico.

Sin embargo, si la función no es polinómica, esta instrucción sólo encontrará una solución. Esto puede comprobarse mediante el siguiente ejemplo. Se

quiere resolver $f(x) = \sin^2 x - \cos x + \mathrm{tg}x = 0$, donde intervienen funciones trigonométricas:

```
>>vpasolve(sin(x)^2-cos(x)+tan(x))
ans =
0.537798851405997707427464179575 31
```

se ha obtenido una única solución aun cuando si se representa dicha función puede observarse que tiene infinitas soluciones

```
>>fplot(sin(x)^2-cos(x)+tan(x))
```

dando lugar a

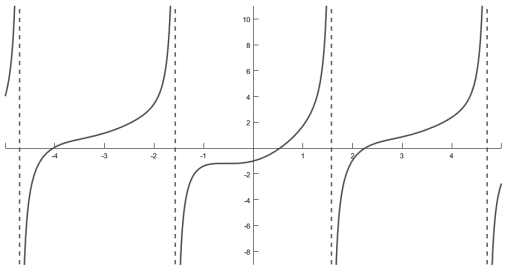


Figura 1.9: Representación gráfica de la función $f(x) = sin^2(x) - cos(x) + tan(x)$

Puede pedirse que encuentre la solución más cercana a cualquier punto, por ejemplo $x = 2$:

```
>>vpasolve(sin(x)^2-cos(x)+tan(x),2)
ans =
2.252172977187969517989595710873 2
```

añadiendo como argumento el punto de partida. También, puede especificarse un intervalo como argumento:

```
>>vpasolve(sin(x)^2-cos(x)+tan(x),[-5,-3])
ans =
-4.031012329991616958935691055685 8
```

Nota 1.5.1 *Destacar cómo hay que introducir la expresión* $\sin^2 x$ *en* Matlab, `sin(x)^2`*; si se quisiera indicar que lo que está elevado al cuadrado es la variable sería:* `sin(x^2)`.

1.5.2 Comando `fzero`

También hay otro comando ya implementado en `Matlab` que resuelve ecuaciones que no pueden resolverse de forma analítica o simbólica, >> `fzero`. Puede hacerlo tanto a partir de un punto inicial como de un intervalo. La función de la que se busca la raíz hay que introducirla como primer argumento y de forma *Anonymus*, es decir, $@(x)f(x)$.

En un método numérico interesa la velocidad con la que accede a la solución. Hay que destacar que el método empleado por la instrucción >> `fzero` es más rápido si se parte de un intervalo inicial que de un punto inicial.

No se sabe qué método emplea, ni se puede acceder al programa interno de esta instrucción pero la ayuda de `Matlab` indica archivos en los que se ha fundado. Comentar al menos que utiliza una combinación de bisección, secante y métodos de interpolación cuadrática inversa.

Debido a eso, lo que sí es necesario en caso de darle un intervalo inicial es que en él se verifique el teorema de Bolzano, o en otras palabras, que la función sea continua y cambie de signo en dicho intervalo, $f \in \mathcal{C}([a,b]), f(a)f(b) < 0 \Rightarrow \exists c \in (a,b), f(c) = 0$. En caso de no ser así, devuelve un mensaje de error

```
>>fzero(@(x)sin(x)^2-cos(x)+tan(x),[-5,-3])
Error using fzero (line 290)
The function values at the interval endpoints must differ in sign.
```

obsérvese en la gráfica de la función representada en la Figura 1.9, que ni siquiera es continua en ese intervalo.

Se puede buscar un intervalo mejor, o ir a lo seguro dando un punto inicial:

```
>>fzero(@(x)sin(x)^2-cos(x)+tan(x),-5)
ans =
-4.7124
```

1.5.3 Ecuaciones polinómicas

Una instrucción que resuelve ecuaciones polinómicas, siempre que el polinomio del que se busquen sus raíces sea dado como un vector formado por los coeficientes ordenados de mayor a menor grado, es

```
>>roots([1 0 0 -2 5])
ans =
-1.0688 + 1.2689i
-1.0688 - 1.2689i
 1.0688 + 0.8212i
 1.0688 - 0.8212i
```

que devuelve las soluciones de $x^4 - 2x + 5 = 0$ indicando que tiene 4 soluciones complejas.

No es un método numérico pero se indica aquí por si puede servir de ayuda al lector en alguna ocasión.

Capítulo 2

Curvas, superficies y derivación de funciones de varias variables

La diferencia entre curva y función es que una curva no siempre tiene que ser una función, puede estar formada por varias funciones. Para que una aplicación se llame función, a cada elemento original x, o elemento del dominio, debe corresponderle a lo sumo una única imagen, $f(x)$.

Por ejemplo, $x^2 + y = 0$ es función ya que $y = -x^2$. Pero, $x + y^2 = 0$ no es una función pues $y = \pm\sqrt{x}$, con lo que a cada valor de x le corresponden dos valores de y. Eso sí, ambas son curvas, concretamente se verá que son parábolas, una vertical y otra horizontal, respectivamente.

Además de trabajar con curvas de la forma $F(x, y) = 0$, si vienen dadas en forma implícita, cuya gráfica estará en $\mathbb{R}^2$ y su dominio son los valores de $x \in \mathbb{R}$ tales que existe $f(x) \in \mathbb{R}$, en este capítulo también se va a trabajar con funciones de varias variables, $z = f(x, y)$, o en forma implícita

$$F(x, y, z) = 0$$

por lo que ahora sus gráficas se representarán en $\mathbb{R}^3$ y el dominio vendrá dado por los $(x, y) \in \mathbb{R}^2$ para los que $f(x, y) \in \mathbb{R}$; es decir, los dominios serán regiones del plano que habrá que describir y dibujar.

A las primeras, con una sola variable independiente, se les llama curvas mientras que a las segundas, con dos variables independientes se les llama superficies. Las primeras se representan en el plano, y su dominio son uniones de intervalos; las segundas son representadas en el espacio y sus dominios son uniones de regiones.

2.1 Dominios de funciones de dos variables

Dada $z = f(x, y) : Dom(f) \subseteq \mathbb{R}^2 \to \mathbb{R}$, se llama dominio de f al conjunto

$$Dom(f) = \{(x, y) \in \mathbb{R}^2 : \ \exists z = f(x, y) \in \mathbb{R}\}$$

Se denomina grafo de $z = f(x, y)$ a las ternas de valores (x, y, z) representadas en $\mathbb{R}^3$

$$Grafo(f) = \{(x, y, z) \in \mathbb{R}^3 : \ (x, y) \in Dom(f), z = f(x, y) \in \mathbb{R}\}$$

Al realizar el análisis de dominios se deberá representar la región del plano resultante. Habrá qué elegir con qué parte del plano separado por curvas hay que quedarse. Se comenzará con un breve recordatorio de algunas cónicas.

2.1.1 Recordatorio de cónicas

Algebráicamente una cónica es una curva en $\mathbb{R}^2$ cuya ecuación general viene dada mediante una ecuación de segundo grado

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

En este texto solo se han considerado cónicas con ejes paralelos a los ejes coordenados, por tanto la ecuación general será:

$$Ax^2 + By^2 + Dx + Ey + F = 0$$

Según los valores de $A, B, C, D, E, F \in \mathbb{R}$ se tendrán diferentes cónicas. A la hora de esbozar rápidamente el gráfico de una cónica siempre es más cómodo partir de la forma canónica (el paso de la ecuación general a la canónica se realiza completando cuadrados). Se ilustra a continuación, como recordatorio, una tabla con los gráficos y ecuaciones canónicas de algunas cónicas.

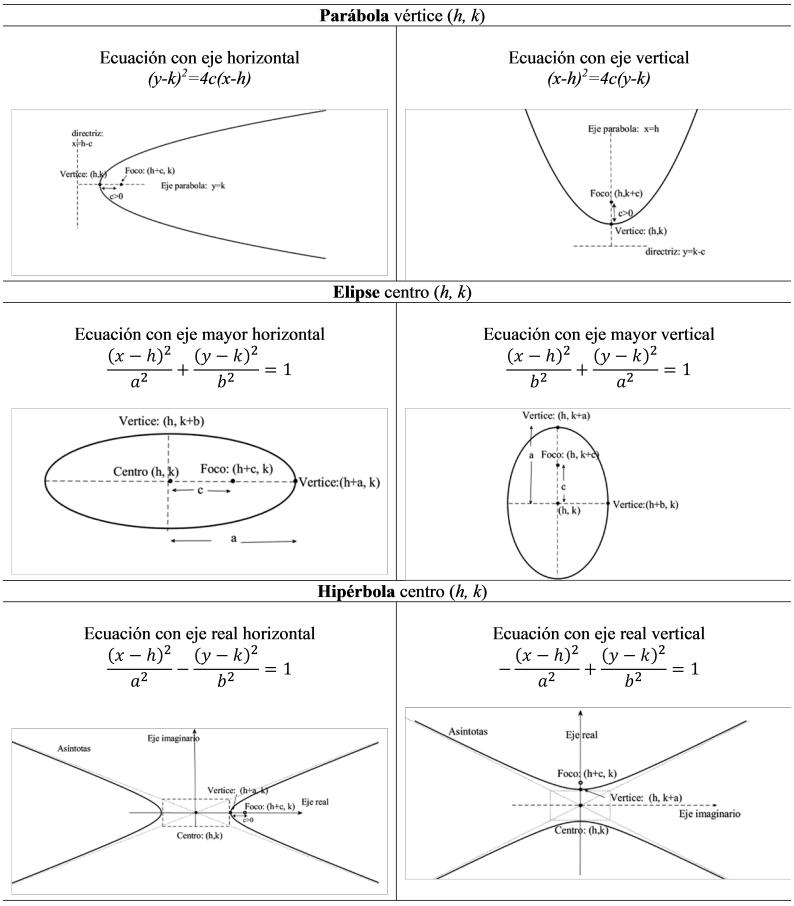


Figura 2.1: Resumen de cónicas

Ejercicio 2.1.1 *Calcula el dominio de* $f(x,y) = \log(8 - x^2 - y^2) + \dfrac{x}{x + y^2}$.

Resolución. Primero se plantea cuáles son los puntos del dominio:

$$Dom f = \left\{(x,y) \in \mathbb{R}^2 : \ 8 - x^2 - y^2 > 0, x + y^2 \neq 0\right\}$$

que son las condiciones para que exista el logaritmo y la fracción. Se busca la zona de $\mathbb{R}^2$ que verifique todas las condiciones:

$$\left.\begin{array}{rl} 8 - x^2 - y^2 & > 0 \\ x + y^2 & \neq 0 \end{array}\right\} \Rightarrow \left.\begin{array}{rl} x^2 + y^2 & < 8 \\ x & \neq -y^2 \end{array}\right\}$$

El siguiente paso será identificar las curvas que intervienen, utilizando la tabla de la Figura 2.1

i) $x^2 + y^2 = 8$ es una circunferencia de $C(0,0)$ y de radio $r = \sqrt{8}$,

ii) $x = -y^2$ es una parábola horizontal de vértice $V(0,0)$ y rama hacia la izquierda.

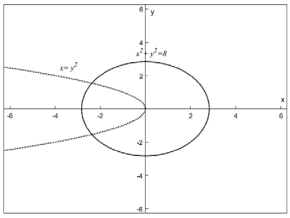


Figura 2.2: Representación de las curvas frontera del dominio

Queda determinar la zona correspondiente a la inecuación, eligiendo puntos del plano, y comprobando su posición respecto a los trozos en los que queda dividido $\mathbb{R}^2$. Se quiere ver cuándo $x^2 + y^2 < 8$, para ello se considera, por ejemplo, el punto $P(2,0)$ y se sustituye: ¿$4 < 8$? Como esto es verdadero, indica que, de las zonas en las que la circunferencia divide al plano, debe cogerse la zona donde está ese punto, es decir, la interior.

Por tanto, el dominio es la región limitada por las curvas en línea discontinua, tanto la circunferencia (ya que la desigualdad es estricta), como la parábola, (pues no debe cogerse al darse en ella la igualdad), e interior a la circunferencia. Si se rellena la región resultante es:

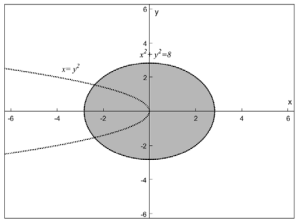


Figura 2.3: Dominio de $f(x,y) = \log(8 - x^2 - y^2) + \dfrac{x}{x + y^2}$

□

Resolución con `Matlab`. Se busca la zona de $\mathbb{R}^2$ que verifique

$$\left.\begin{array}{rl} 8-x^2-y^2 & >0 \\ x+y^2 & \neq 0 \end{array}\right\} \Rightarrow \left.\begin{array}{rl} x^2+y^2 & <8 \\ x & \neq -y^2 \end{array}\right\}$$

representando las curvas que intervienen, Figura 2.3, mediante la instrucción

```
>>syms x y,ezplot(x==-y^2),hold on,ezplot(x^2+y^2==8)
```

Para analizar la zona basta considerar un punto y comprobar si cumple la relación. En este caso, por ejemplo, se sustituye el punto $(0.5, 0.5)$

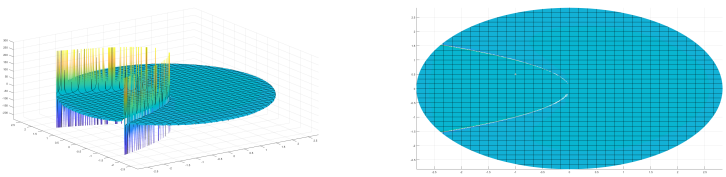
```
>>subs(x^2+y^2<=8,[x,y],[0.5,0.5])
ans =
1/2 <= 8
```

como es cierta la desigualdad se toma la zona donde está el punto.

El dominio es la proyección sobre el plano XY, Figura 2.4(b). `Matlab` permite girar los grafos usando la herramienta del menú *Tools* y seleccionando a continuación *Rotate 3D*. Dibujando primero la superficie

```
>>f(x,y)=log(8-x^2-y^2)+x/(x+y^2);ezsurf(f)
```

Se aconseja etiquetar los ejes antes de hacerlo girar. Seleccionando en el desplegable del menú *Insert* la opción de *X Label* y/o *Y Label*, se coloca el cursor en el eje correspondiente y se escribe el nombre del eje o variable. Con el ratón situado sobre la figura se gira hasta poner el eje OY en vertical y el eje OX en horizontal o bien con el botón derecho del ratón aparece un cuadro con opciones para acceder a algunas de las proyecciones, Figura 3.26



(a) Función $f(x, y)$ (b) Proyección en el plano XY

Figura 2.4: Representación de $f(x, y) = \log(8 - x^2 - y^2) + \frac{x}{x+y^2}$

⊠

Ejercicio 2.1.2 *Dada* $f(x,y)=\sqrt{x^2+4y^2-1}+\frac{\log(-4x^2-y^2+4)}{x^2-y-2x}$,

a) Analiza el dominio de definición.

b) Representa dicho dominio.

Resolución.

a) Para que la función exista debe cumplirse:

$$Dom f=\left\{(x,y)\in\mathbb{R}^2:\ x^2+4y^2-1\geq 0,-4x^2-y^2+4>0,x^2-y-2x\neq 0\right\}$$

lo que con un sistema sería

$$\left.\begin{array}{rl} x^2+4y^2-1 & \geq 0\\ -4x^2-y^2+4 & >0\\ x^2-y-2x & \neq 0\end{array}\right\}\Rightarrow\left.\begin{array}{rl} x^2+\frac{y^2}{1/4} & \geq 1\\ 4x^2+y^2 & <4\\ (x-1)^2-y & \neq 1\end{array}\right\}\Rightarrow\left.\begin{array}{rl} x^2+\frac{y^2}{1/4} & \geq 1\\ x^2+\frac{y^2}{4} & <1\\ (x-1)^2 & \neq y+1\end{array}\right\}$$

Si se observan las cónicas que entran en juego, (es decir, primero fijarse en las curvas, las igualdades, y luego ya estudiar las zonas), se tiene:

i) $x^2+\frac{y^2}{1/4}=1$ es una elipse horizontal de $C(0,0)$, de semieje mayor $a=1$, semieje menor $b=1/2$, y por tanto $c=\frac{\sqrt{3}}{2}$,

ii) $x^2+\frac{y^2}{4}=1$ es una elipse vertical de $C(0,0)$, semieje mayor $a=2$ y semieje menor $b=1$,

iii) $y+1=(x-1)^2$ es una parábola vertical con vértice $V(1,-1)$ que mira hacia arriba.

Dibujando las tres curvas:

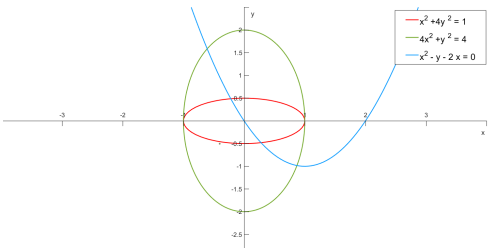


Figura 2.5: Representación de las curvas del dominio

Ahora queda determinar las zonas correspondientes a cada una de las inecuaciones, para eso hay que tomar puntos del plano, y comprobar las posiciones en cada región de las que queda dividido $\mathbb{R}^2$.

- Zona 1: se quiere ver cuándo $x^2+\frac{y^2}{1/4} \geq 1$, para ello considerar por ejemplo el punto $P(0,0)$ y se sustituye: ¿$0 \geq 1$? Como esto es falso, indica que, de las zonas en las que la elipse divide al plano, debe cogerse la zona donde no está ese punto.

- Zona 2: ver cuándo $x^2 + \frac{y^2}{4} < 1$. De nuevo considérese el punto $P(0,0)$. Sustituyendo: ¿$0 < 1$? Evidentemente es cierto, así que se toma la zona donde está ese punto, es decir, el interior de la elipse. Teniendo cuidado de que la elipse hay que trazarla discontinua, ya que al tener un menor estricto, la curva no puede entrar en el dominio.

- Zona 3: en este caso se necesita que $y + 1 \neq (x-1)^2$, por lo tanto, basta con dibujar discontinua esta parábola, ya que lo que no puede darse es la igualdad.

  Si se rellenan estas zonas en el dibujo:

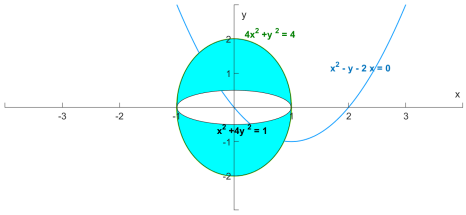


Figura 2.6: Dominio de $f(x,y) = \sqrt{x^2+4y^2-1} + \frac{\log(-4x^2-y^2+4)}{x^2-y-2x}$

□

Resolución con `Matlab`. El planteamiento del ejercicio es el mismo. `Matlab` ayudará a resolver inecuaciones, o mejor, representaciones gráficas ya que desigualdades de dos variables no las trabaja de una forma sencilla de interpretar por el usuario.

Se busca la zona de $\mathbb{R}^2$ que verifique

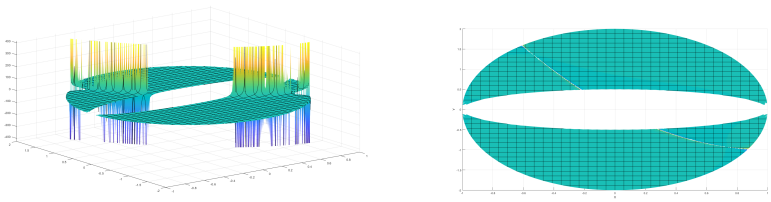
$$\left.\begin{array}{rl} x^2+4y^2-1 & \geq 0 \\ -4x^2-y^2+4 & > 0 \\ x^2-y-2x & \neq 0 \end{array}\right\} \Rightarrow \left.\begin{array}{rl} x^2+\frac{y^2}{1/4} & \geq 1 \\ 4x^2+y^2 & < 4 \\ y+1 & \neq (x-1)^2 \end{array}\right\} \Rightarrow \left.\begin{array}{rl} x^2+\frac{y^2}{1/4} & \geq 1 \\ x^2+\frac{y^2}{4} & < 1 \\ y+1 & \neq (x-1)^2 \end{array}\right\}$$

para ello se representan todas las curvas que intervienen, Figura 2.5,

```
>>syms x y
>>ezplot(y+1-(x-1)^2),hold on,ezplot(x^2+y^2/4-1),hold on,ezplot(x^2+4*y^2-1)
```

Para obtener el dominio sobre el plano XY basta rotar la figura como se ha indicado en el Ejercicio 2.1.1, dando lugar la Figura 2.7(b)

```
>>f(x,y)=sqrt(x^2+4*y^2-1)+log(- 4*x^2 - y^2 + 4)/(x^2 -y -2*x);fsurf(f)
```



(a) Superficie $f(x,y)$ (b) Proyección sobre XY

Figura 2.7: Representación de $f(x,y)=\sqrt{x^2+4y^2-1}+\frac{\log(-4x^2-y^2+4)}{x^2-y-2x}$

⊠

Ejercicio 2.1.3 *Halla el dominio de* $f(x,y)=\sqrt{\dfrac{2x^2+y^2-6x-2}{y-5+x^2}}$.

Resolución. Los pares de valores del dominio son aquellos que cumplen:

$$Domf=\left\{(x,y)\in\mathbb{R}^2:\ \frac{2x^2+y^2-6x-2}{y-5+x^2}\geq 0, y-5+x^2\neq 0\right\}$$

Por tanto las condiciones a analizar son las siguientes

$$\left.\begin{array}{rl} \dfrac{2x^2+y^2-6x-2}{y-5+x^2} & \geq 0 \\ y-5+x^2 & \neq 0 \end{array}\right\}$$

Para seguir leyendo, inicie el proceso de compra, click aquí