Introducción a MATLAB

Filánder Sequeira Chavarría
Esteban Segura Ugalde
Óscar Salas Huertas

EDITORIAL UCR

Introducción a MATLAB

Filánder Sequeira Chavarría
Esteban Segura Ugalde
Óscar Salas Huertas

EDITORIAL UCR

519.4
S479i3 Sequeira Chavarría, Filander, 1986-
Introducción a MATLAB / Filander Sequeira Chavarría, Esteban Segura Ugalde, Oscar Salas Huertas. – 1. edición. – San José, Costa Rica : Editorial UCR, 2020.
1 recurso en línea (xiv, 238 páginas) : ilustraciones (algunas a color), gráficos (principalmente a color), digital, archivo de texto, PDF ; 9.3 MB

ISBN 978-9968-46-859-6

1. ANÁLISIS NUMÉRICO – PROCESAMIENTO DE DATOS. 2. MATLAB (PROGRAMAS PARA COMPUTADORA). I. Segura Ugalde, Esteban, 1984- , autor. II. Salas Huertas, Oscar, 1977- , autor. III. Título.

CIP/3510

CC.SIBDI.UCR

Edición aprobada por la Comisión Editorial de la Universidad de Costa Rica.
Primera edición impresa: 2020.
Primera edición digital (PDF): 2020.

Editorial UCR es miembro del Sistema Editorial Universitario Centroamericano (SEDUCA), perteneciente al Consejo Superior Universitario Centroamericano (CSUCA).

Corrección filológica: *Euclides Hernández P.* • Revisión de pruebas: *Gabriela Fonseca A.* • Diseño y diagramación: *los autores*
Diseño de portada: *Raquel Fernández C.* • Control de calidad de la versión impresa: *Abraham Ugarte S.* • Realización de PDF: *Hazel Aguilar B.* • Control de calidad de la versión digital: *Elisa Giacomin V.*

Edición digital de la Editorial Universidad de Costa Rica. Fecha de creación: julio, 2020
Universidad de Costa Rica. Ciudad Universitaria Rodrigo Facio. San José, Costa Rica.

Apdo. 11501-2060 • Tel.: 2511 5310 • Fax: 2511 5257 • administracion.siedin@ucr.ac.cr • www.editorial.ucr.ac.cr

Contenido

CAPÍTULO 1

Preliminares

 MATLAB® (MATrix LABoratory) es un programa especializado en la ejecución de cálculos numéricos que involucran vectores y matrices con entradas reales o complejas. Lo anterior lo hace una de las herramientas más utilizadas para realizar investigaciones afines a la computación científica. Al respecto, este software posee un conjunto de paquetes o librerías, denominados toolboxes, para diferentes áreas de estudio, como por ejemplo: matemática, física, química, ingeniería, entre otras. En particular, para la rama de matemática, se encuentran toolboxes para ecuaciones diferenciales, optimización, estadística, procesamiento de imágenes, por mencionar algunos.

MATLAB® tiene versiones para los sistemas operativos Windows y UNIX, con el inconveniente de que es un paquete computacional no gratuito y disponible solo en inglés. A pesar de ello, en muchas universidades, laboratorios y centros de investigación, representa la principal herramienta de desarrollo de sus quehaceres, en virtud de su facilidad de sintaxis, sus continuas actualizaciones y mejoras, así como su eficiencia al realizar una gran cantidad de cálculos rápidamente que aprovechan las arquitecturas existentes.

Es importante resaltar la existencia de programas gratuitos que siguen la misma línea de MATLAB®. Estas opciones resultan ser atractivas cuando se requieren realizar cálculos no muy especializados o complejos y son bastante útiles cuando se trabaja en cursos de docencia. Entre las principales opciones se destacan:  Scilab,  GNU Octave,  FreeMat y  Maxima. En particular, los autores recomiendan la utilización de Scilab, debido a que es muy compatible con MATLAB®, razón por la cual en el Apéndice C se presenta una breve descripción de este.

1.1 Interfaz gráfica

MATLAB® brinda al usuario una interfaz gráfica bastante simple y amigable. En efecto, al iniciarlo por primera vez (luego de la instalación) su apariencia será como se muestra en la Figura 1.1, donde se aprecia el menú principal que organiza las herramientas por medio de pestañas. En este caso, se ilustran tres pestañas: HOME, PLOTS y APPS. El contenido de las pestañas puede ocultarse mediante el ícono que aparece en el margen superior derecho.

Por otro lado, en la Figura 1.1, se observan múltiples ventanas, las cuales se describen brevemente en esta sección. Estas ventanas pueden ser arrastradas según la comodidad del usuario. Más aún, se pueden separar (Undock) o unir (Dock) por medio de las opciones Undock y Dock, que se encuentran en el menú de acciones de la parte superior derecha de cada ventana (véase Figura 1.2).

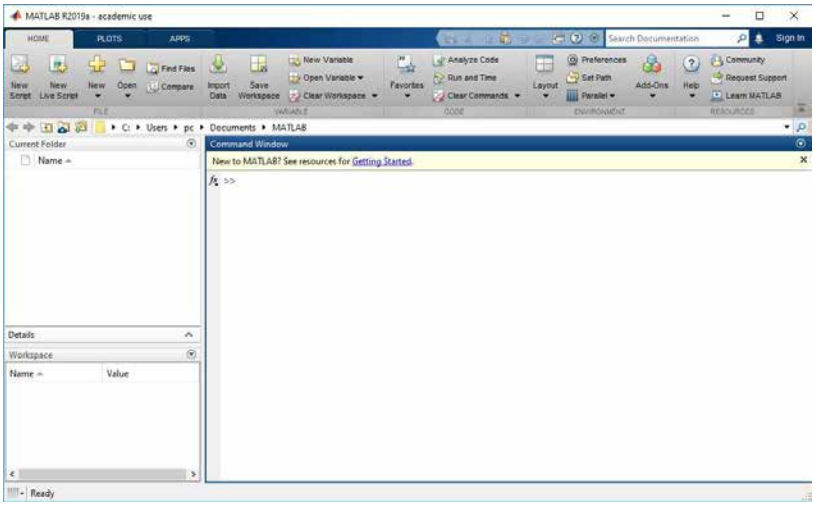


Figura 1.1 Interfaz inicial de MATLAB® R2019a

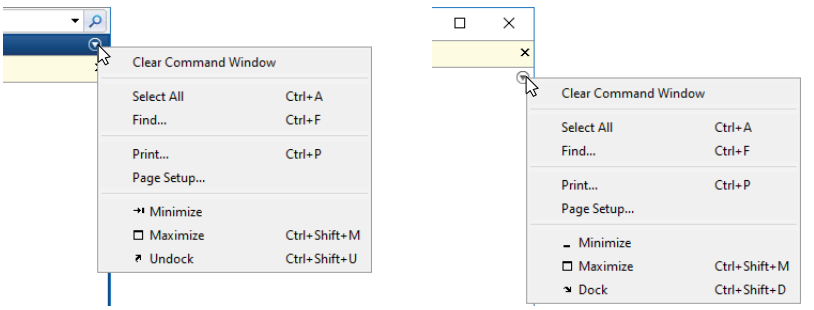


Figura 1.2 Menú de acciones de una ventana

La opción ↗ Undock le permite a la ventana separarse (independizarse) de la ventana principal, denominada Desktop, mientras que la opción ↘ Dock une la ventana a Desktop.

Así, la primera recomendación es personalizar el Desktop para sentirse cómodo al interactuar con el programa. Por ejemplo, algunos usuarios prefieren tener las ventanas independientes unas de otras, mientras que otros, como los autores, una única ventana. MATLAB® trae algunas configuraciones de ventana, las cuales se pueden acceder en la opción Layout de la pestaña HOME (véase Figura 1.3). La Figura 1.1 previa, muestra la configuración Default que es la recomendada por MATLAB®.

Figura 1.3 Configurar las ventanas de MATLAB®

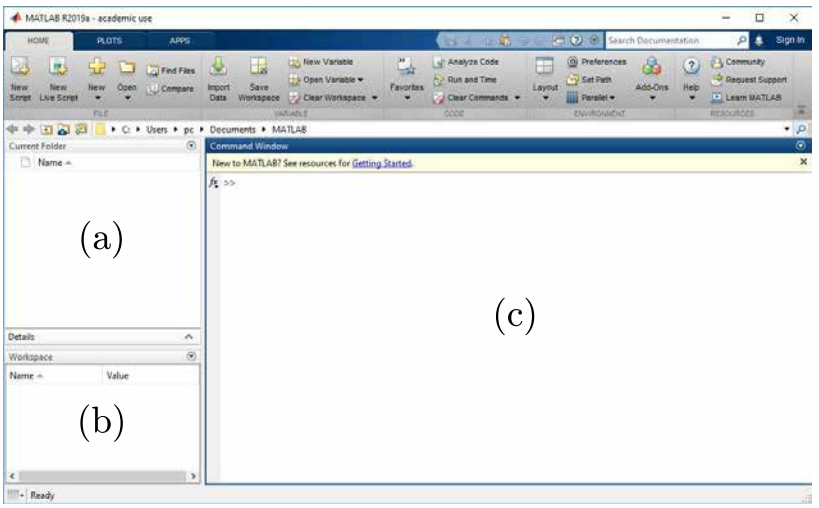


Figura 1.4 Configuración Default de MATLAB®

En la Figura 1.4, se han enumerado las ventanas de la siguiente manera: **(a)** corresponde a **Current Folder**, **(b)** a **Workspace** y **(c)** a **Command Window**, las cuales permiten una interacción amigable. Sin embargo, adicional a estas ventanas, los autores consideran de gran relevancia incorporar la ventana **Editor**, que no se muestra por defecto en un principio. A continuación se realiza una breve descripción de cada una de estas ventanas.

1.1.1 Command Window y Workspace

La ventana Command Window (o ventana de comandos) es donde se ejecutan secuencialmente los comandos de MATLAB®. En esta aparece el **prompt** (>>), el cual se utiliza para indicar que el programa está preparado para recibir instrucciones. Si no aparece, se debe a que MATLAB® está ocupado (Busy) realizando alguna instrucción previa, por lo que no realizará nuevas instrucciones hasta haber finalizado la anterior (véase Figura 1.5).

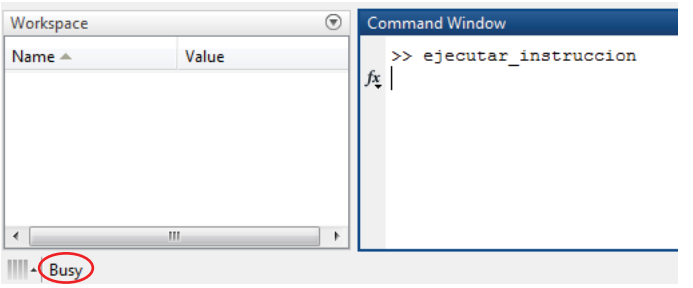


Figura 1.5 Instrucción en proceso

Se debe tener en cuenta que cuando se presiona la tecla de entrada Enter en la ventana de comandos, se ejecutan las instrucciones que se han digitado. Para realizar un cambio de línea, sin provocar que MATLAB® realice la ejecución, se debe presionar la tecla Shift junto con la tecla de entrada para lograrlo, es decir, Shift + Enter. El siguiente ejemplo ilustra un trabajo en la ventana de comandos.

Ejemplo 1.1 Escriba en la ventana de comandos la instrucción:

```
I = eye(6)
```

y presione la tecla de entrada. El resultado es la siguiente matriz:

```
I =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

Observación: Del ejemplo anterior se puede notar lo siguiente:

1. Al ejecutar `I = eye(6)` se crea una matriz identidad de orden 6 con el nombre `I`. Esto se debe a que en los toolboxes de MATLAB®, ya se encuentra reservada la palabra `eye` para un programa que genera matrices identidades con el orden que se desee, el cual se escribe entre paréntesis redondos (véase Cuadro 2.1). A estos programas se les denominan: **funciones** o **rutinas**. Adicional a las funciones ya predefinidas en MATLAB®, en el Capítulo 3 se describe como crear algunas de ellas.

2. Palabras no reservadas por MATLAB® provocan un mensaje de error. En efecto, al ejecutar `hola` aparece el mensaje de error:

   ```
   Undefined function or variable 'hola'.

   Did you mean:
   >> hold
   ```

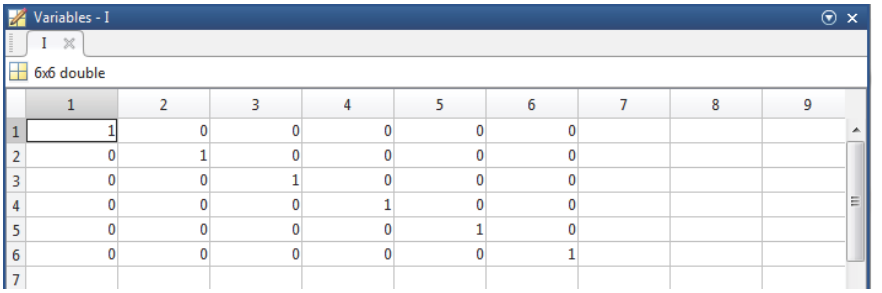
   el cual hace referencia a que `hola` no se reconoce, y en algunos casos sugiere alguna palabra reservada que se me asemeja en sintaxis.

3. Se debe tener en cuenta que MATLAB® **es sensible a mayúsculas**, es decir, no se obtiene el mismo resultado al escribir las instrucciones: `eye`, `EYE`, `Eye` o `eYe`.

4. Nótese que la matriz `I` se muestra en la ventana de comandos. En el caso de que no sea necesario que aparezca, pero sí crearla, basta con agregar un punto y coma (`;`) al final de la instrucción, es decir, `I = eye(6);`. En general, solo cuando se agrega un punto y coma al final de una instrucción, MATLAB® previene que el **resultado no se debe mostrar** en la ventana de comandos.

5. Luego de ingresar una instrucción en la ventana de comandos, la misma se puede volver a ejecutar por medio de las teclas:  y  . De hecho, con estas teclas se puede recorrer el historial de ejecuciones realizadas en sesiones previas.

Por otro lado, nótese que en la ventana de Workspace se muestra la `I` que dice además *6x6 double*. Esto debido a que la ventana Workspace (o de memoria de trabajo) contiene la información sobre todas las variables que se hayan definido durante la sesión actual de MATLAB®. En el caso de las matrices, también se pueden ver y modificar sus entradas simplemente con dar *doble clic* sobre su nombre, como se muestra en la Figura 1.6.



	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0			
2	0	1	0	0	0	0			
3	0	0	1	0	0	0			
4	0	0	0	1	0	0			
5	0	0	0	0	1	0			
6	0	0	0	0	0	1			
7									

Figura 1.6 Entradas de la matriz `I` a través de Workspace

Los datos que se tengan en el Workspace no se guardan automáticamente. De hecho, al finalizar la sesión de MATLAB® estos son borrados. Más adelante se aclara cómo guardar estos datos para ser utilizados en sesiones posteriores (véase Sección 2.2.4). Si se desea borrar una variable específica del Workspace sin cerrar la sesión, por ejemplo la variable `I` creada anteriormente, se puede utilizar la instrucción:

```
clear I
```

En el caso de necesitar borrar todas las variables (es decir, limpiar el Workspace), se puede ejecutar el comando `clear all`. Nótese que `clear all` no borra el contenido de la ventana de comandos. Esto se puede realizar por medio de la instrucción `clc`, la cual tampoco afecta el Workspace.

Para ingresar varias instrucciones simultáneamente, en la ventana de comandos se usa la coma sencilla (`,`) o el punto y coma (`;`) para separar cada instrucción. Esto se ilustra en el siguiente ejemplo.

Ejemplo 1.2 Escriba en la ventana de comandos las instrucciones:

```
A = rand(6),  B = rand(6),  B*A
```

y presione la tecla de entrada.

Observación: Del ejemplo anterior se puede notar lo siguiente:

1. La función `rand(n)` crea una matriz de tamaño `n`×`n`, con entradas generadas al azar entre 0 y 1. Por lo tanto, los resultados obtenidos para `A` y `B`, pueden variar dado que se generaron de manera aleatoria.

2. El resultado del ejemplo muestra tres matrices en la ventana de comandos. A las dos primeras se les asignó (por medio del símbolo: `=`) el

nombre de `A` y `B`, respectivamente, mientras que a la tercera correspondiente a la operación `B*A`, cuyo nombre no se especifica, se almacena por defecto en una variable de nombre `ans` (diminutivo de answer). Se recomienda siempre establecer nombres para las variables, dado que los valores en `ans` se pueden reemplazar con facilidad. Para mayor detalle véase Sección 1.3.

3. La tercera instrucción `B*A`, corresponde al producto usual entre las matrices `B` y `A`, por lo que las dimensiones de las mismas deben ser las adecuadas. En caso contrario se produce un error de ejecución. En efecto, al efectuar las instrucciones `A = rand(4), B = rand(6), B*A`, se produce el mensaje de error:

```
Error using *
Inner matrix dimensions must agree.
```

4. Las entradas de las matrices del ejemplo presentan 4 decimales, lo cual es una representación por defecto. Por ejemplo, al ingresar al matriz `C = rand(4,6)` se obtiene:

```
C =

    0.7487    0.5341    0.6787    0.1476    0.9296    0.8790
    0.8256    0.0900    0.4952    0.0550    0.6967    0.9889
    0.7900    0.1117    0.1897    0.8507    0.5828    0.0005
    0.3185    0.1363    0.4950    0.5606    0.8154    0.8654
```

Aunque aparezcan solo 4 decimales en la ventana de comandos, es importante mencionar que MATLAB® almacena los valores numéricos con un máximo aproximado de 16 cifras, lo cual se conoce como *doble precisión*. Si se desean ver todas las cifras de los valores numéricos, basta escribir una vez la instrucción `format long`, o bien, `format long e`.

1.1.2 Editor y Current Folder

Tal y como se menciona previamente, al cerrar la sesión de MATLAB® todas las variables almacenadas en el Workspace son eliminadas, por lo que no podrán ser usadas en futuras sesiones. En el caso de necesitar continuar con la información de una sesión previa, se recomienda el uso de la ventana denominada **Editor**. Esta no aparece en la configuración por defecto.

Para abrir la ventana del Editor, se ingresa en la opción  New Script de la pestaña HOME. Este se muestra en la posición (d) de la Figura 1.7.

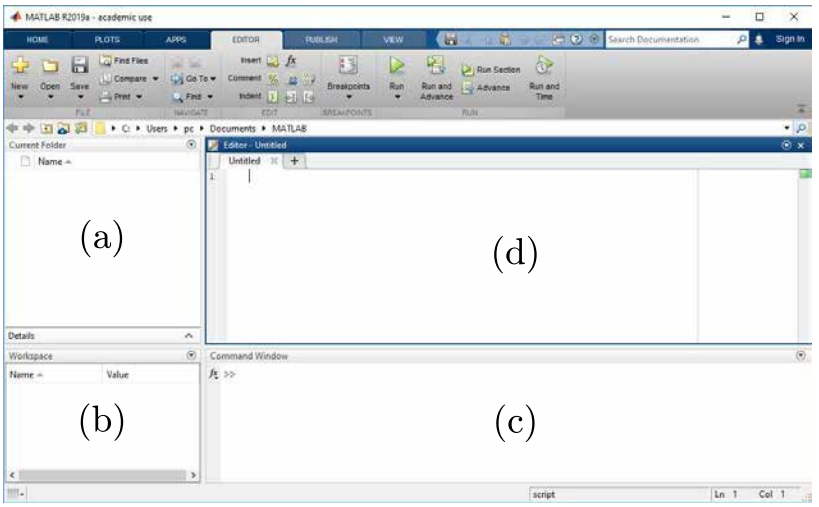


Figura 1.7 Interfaz gráfica de MATLAB® con el Editor

El Editor es una herramienta muy útil, ya que permite ejecutar y preservar archivos con fragmentos de código de MATLAB®, similar a guardar un archivo de texto. Estos archivos se guardan con extensión ***.m** y la interfaz de **Current Folder** muestra el directorio actual donde MATLAB® está considerando los archivos a ejecutar. En el Editor se puede presionar la tecla Enter sin que esto ejecute las instrucciones, como ocurre en la ventana de comandos. Para ejecutar todas las instrucciones en el Editor basta con oprimir la tecla F5 o dar *clic* sobre la opción Run de la pestaña EDITOR.

Ejemplo 1.3 (Primer archivo *.m)

a) Escriba en el Editor el siguiente fragmento de código:

```
clc, clear all
n = 50;
I = eye(n);
E = I(:, n:-1:1);
spy(I + E);
disp('Hola Mundo');
```

b) Utilice la opción Save para guardar el archivo con el nombre de `primerArchivo.m`.

c) Ejecute este código, presionando la tecla F5. Si la ejecución fue exitosa se obtiene el resultado de la Figura 1.8. Es posible que en lugar del resultado de la Figura 1.8, se obtenga un mensaje con opciones como el de la Figura 1.9. La aparición de este mensaje se debe a que el archivo `primerArchivo.m` se guardó en un directorio distinto al Current Folder. En efecto, en el Current Folder se muestra la dirección `C:\Users\pc\Documents\MATLAB` pero la ubicación del archivo fue en `C:\Users\pc\Desktop`. Así, la ventana de la Figura 1.9 ayuda a cambiar el Current Folder donde se encuentra el archivo por medio de la opción **Change Directory**. Este mensaje no aparecerá nuevamente, mientras no se modifique el Current Folder.

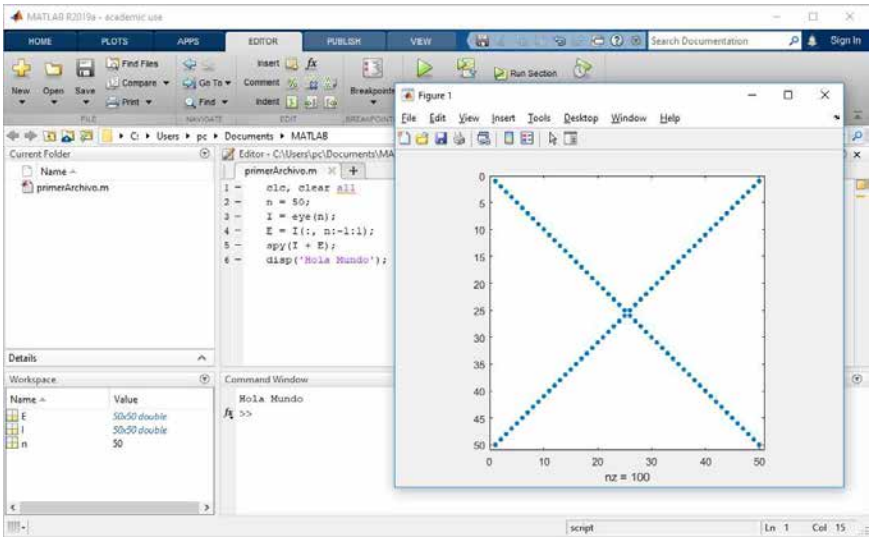


Figura 1.8 Resultado del Ejemplo 1.3

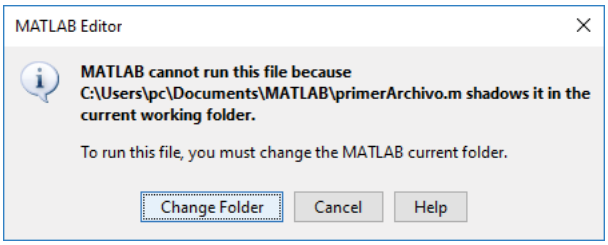


Figura 1.9 Cambiar el directorio actual de trabajo (Current Folder)

Otra ventaja del Editor consiste en que se pueden ingresar cambios de línea, espacios y tabulares para que el texto quede más amigable a vista del lector. Además, MATLAB® reserva el símbolo % para introducir **comentarios** dentro

de los códigos. Estos consisten de textos explicativos que no son interpretados por MATLAB®. Lo usual es usar pequeños mensajes significativos que ubiquen al usuario sobre lo que hace una o varias instrucciones. Todo lo anterior permite que el usuario personalice sus códigos, por ejemplo, algunos programadores inician sus rutinas ingresando su nombre y fecha de creación como se muestra:

```
% Autor: Filánder Sequeira Chavarría
% Fecha: 19 de Julio de 2017
```

Ejercicio 1.1 Considere el código de la parte *a*) del Ejemplo 1.3, y agregue comentarios que expliquen cada una de las líneas allí presentes.

Observación: Para comentar bloques de instrucciones, es decir, para comentar más de una línea a la vez, se puede usar la opción Comment de la pestaña HOME, o bien, presionar el botón derecho del ratón y oprimir la opción Comment. Análogamente, en ambas versiones existe la opción Uncomment.

1.2 La ayuda de MATLAB®

MATLAB® dispone de una detallada documentación que guía al usuario sobre un adecuado uso de sus diversos componentes, como lo son: algoritmos, operaciones, variables, estructuras de datos, entre otras (véase Figura 1.10). En esta, se puede encontrar explicaciones de funciones propias de MATLAB®, donde se presenta: su sintaxis, descripción, ejemplos, referencias bibliográficas, funciones alternativas relacionadas, por mencionar algunas.

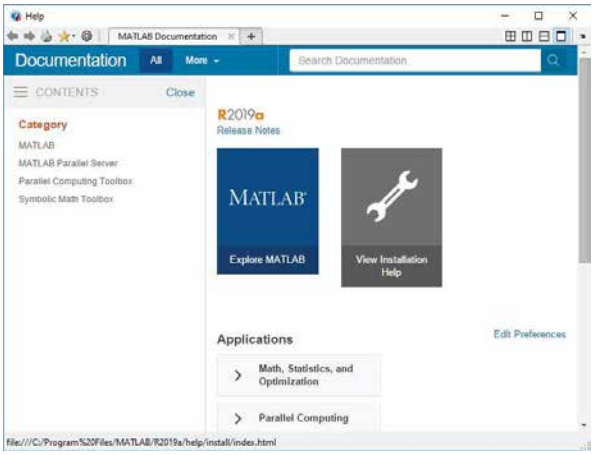


Figura 1.10 Ventana de ayuda de MATLAB®

A continuación se describen las principales formas de acceder a la ayuda de MATLAB®:

1. La opción Help de la pestaña HOME, abre la ventana de la ayuda, donde a su izquierda se presenta un menú de categorías denominado CONTENTS. En este menú se puede navegar para buscar algún comando de interés, cuando el nombre de este no se conoce. Por ejemplo, al ingresar en el menú las opciones: MATLAB, Mathematics, Elementary Math y Discrete Math (véase Figura 1.11) se puede apreciar la función denominada `factor`. Al dar *clic* sobre ella se muestra la documentación de este comando, el cual determina los factores primos de un número natural (véase Figura 1.11).

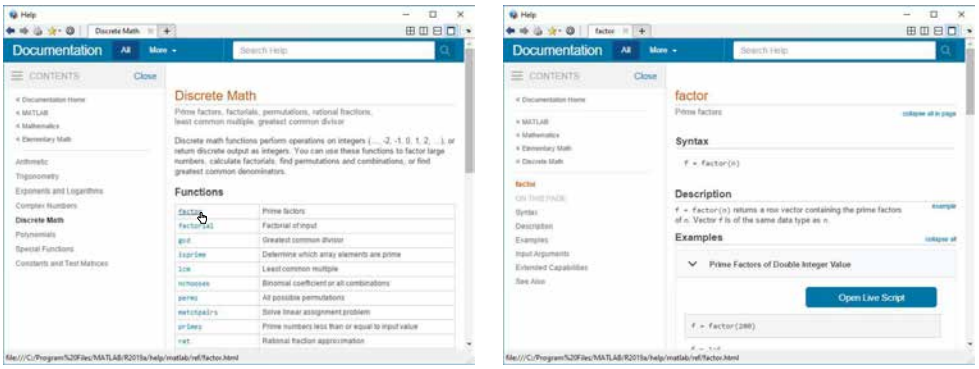


Figura 1.11 Búsqueda en la Ayuda

Es importante aclarar que siempre es posible acceder a la Ayuda, mediante el ícono Searc, que se encuentra en el margen superior derecho de la ventana (véase Figura 1.12).

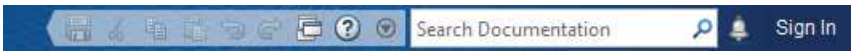


Figura 1.12 Campo para buscar en la Ayuda

2. Otra opción es utilizar el campo de búsqueda denominado *Search Documentation* (véase Figura 1.12) para obtener información sobre una función a través de alguna palabra clave, tal y como se muestra en la Figura 1.13.

3. Además de las opciones anteriores, también se puede accesar a la ayuda desde la ventana de comandos o el Editor. Para esto, se debe digitar el nombre del comando y seguidamente presionar la tecla F1. Esto se ilustra en la Figura 1.14 para el comando `factor` en la ventana de comandos (izquierda) y el Editor (derecha).

Figura 1.13 Campo para buscar en la Ayuda

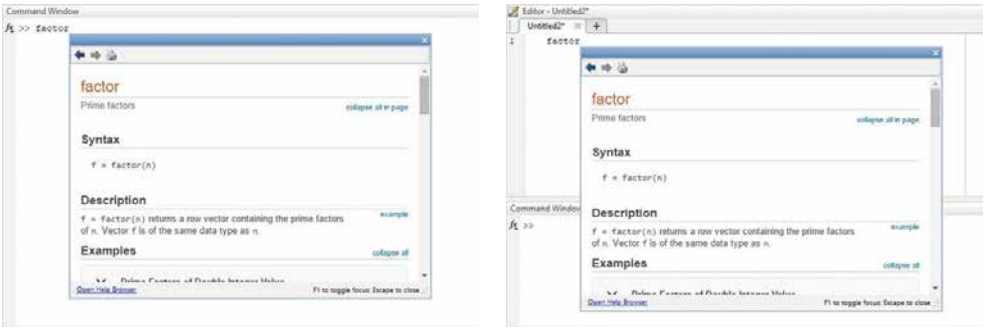


Figura 1.14 Documentación del comando `factor` usando la tecla F1

4. Finalmente, podemos desplegar la ayuda en la misma ventana de comandos por medio la instrucción `help`, o bien, `help` seguido del nombre de un comando. En la Figura 1.15, se muestra lo que obtiene al ejecutar `help factor`. Además, reemplazando `help` por `helpwin` la información se muestra en una ventana aparte. Por otro lado, la instrucción `help` aporta información básica del comando consultado, en comparación con la que se encuentra en la documentación Help.

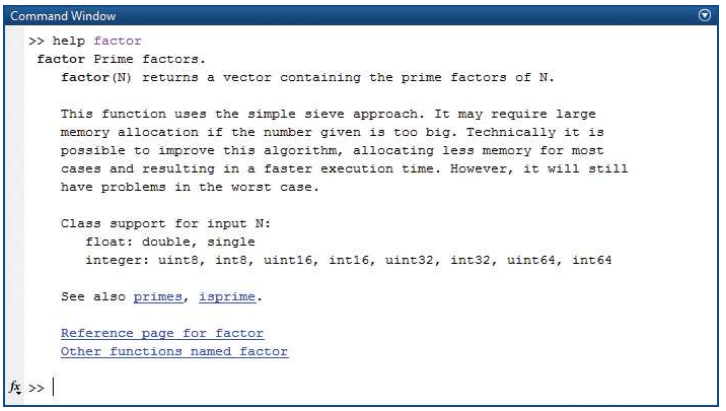


Figura 1.15 Resultado de la instrucción `help factor`

1.3 Variables y operaciones

Cuando se realiza la operación de asignación `I = eye(6)` del Ejemplo 1.1, se crea una variable llamada `I` que almacena el valor `eye(6)` (en este caso, la matriz identidad de 6×6). Así, el **operador de asignación** es denotado por el signo de igualdad (=). En otras palabras, cuando se usa la expresión `a = c`, se debe entender que *se le asigna a la variable* `a`*, el valor contenido en la variable* `c`. Es importante notar que este operador no es conmutativo, es decir, la expresión `eye(6)= I` produce un error, dado que no se puede asignar el valor de `I` al comando `eye(6)`.

En general, cuando se trabaja con un lenguaje de programación (como el que brinda MATLAB®), la definición de variables (o bien, la asignación de nombres a valores) se convierte en una gran ayuda. Por ejemplo, suponga que es necesario utilizar varias veces la matriz identidad de orden 6, para lo cual es posible usar la instrucción `eye(6)`. Sin embargo, cada vez que se emplee `eye(6)`, MATLAB® lleva acabo el proceso de construcción de esta matriz, el cual podría requerir un tiempo significativo de cálculo. Esto se puede evitar, al asignarle el nombre de `I`, de la forma `I = eye(6)`, donde ahora se construye primero la matriz y luego esta se guarda en `I`. Así, utilizar `I` varias veces no conlleva un gasto de tiempo de construcción, dado que la matriz ya ha sido creada y almacenada.

De acuerdo con la discusión anterior, el uso y la correcta definición de variables será de gran ayuda. En particular, a la hora de escribir el nombre a una variable se deben tener en cuenta las siguientes reglas:

- Iniciar los nombres con letras y no con números. Por ejemplo, `x1` es un nombre correcto, pero `4ever` no lo es.
- Utilizar nombres significativos. Por ejemplo, si se va a almacenar una fecha, se pueden utilizar variables con nombres como: `dia`, `mes` y `anho`, o bien, en inglés: `day`, `month` y `year`, ya que indican claramente que están guardando.
- No incluir: espacios, signos de admiración e interrogación, delimitadores y operaciones ((), [], {}, +, −, ∗, /), ni caracteres especiales o acentos (á, é, í, ó, ú, ñ, ç, &, $, #, @), etc. El guión bajo (_) si es permitido.
- Recordar que hay diferencia entre letras mayúsculas y minúsculas. Por ejemplo, la variable `x` es distinta a la variable `X`.

- Por conveniencia, para los nombres de los vectores y escalares se utilizan letras minúsculas, mientras que los nombres de las matrices serán con letras mayúsculas.

A diferencia de muchos lenguajes de programación, para crear una variable en MATLAB® no es necesario definir previamente el tipo de dato a almacenar. En efecto, MATLAB® reconoce automáticamente el tipo de variable, el cual puede ser modificado cuando se desee. Por ejemplo, ingrese en la ventana de comandos las siguientes instrucciones:

```
salario = 10000;
longitud = 12.7;
autor = 'Óscar';
valor_de_verdad = true; % Revisar: help true
A = rand(2);
```

Es claro que la variable `salario` corresponde a un entero, mientras que `longitud` es un valor decimal (double o punto flotante). Usualmente en otros lenguajes de programación se debe definir previamente el tipo de variable, como por ejemplo, `int salario = 10000;` y `double longitud = 12.7;`. A pesar de no hacerlo, MATLAB® lo interpreta correctamente tal como se aprecia en la Figura 1.16. Similarmente, la variable `A` que representa una matriz 2×2 es representada por cuatro valores decimales. Por otro lado, las variables `autor` y `valor_de_verdad` corresponden a una cadena de caracteres (véase Sección 2.4) y un valor booleano (véase Sección 6.1), respectivamente.

Figura 1.16 Ejemplo de variables en Workspace

Por otro lado, en el Workspace es posible ver información básica de las variables que se han creado, en particular, su tipo y cantidad de memoria requerida. Para más opciones, se puede oprimir el botón derecho del ratón sobre la barra superior del Workspace, tal y como se muestra en la Figura 1.17

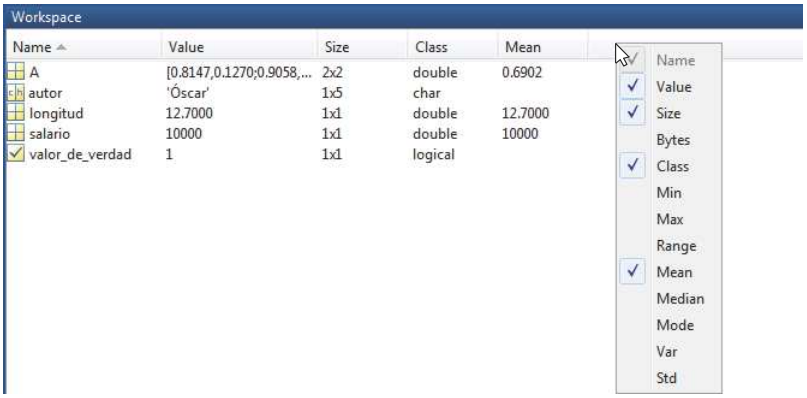


Figura 1.17 Opciones del Workspace

Observación: Los números pueden escribirse utilizando la notación científica, la cual corresponde a representar los valores mediante una potencia de diez. Por ejemplo, en lugar de ingresar `0.00014` se puede ingresar de la forma `1.4e-4`, donde la `e` en medio de los dos números denota la potencia de diez, es decir, `1.4e-4` $= 1.4 \times 10^{-4}$. Similarmente, `1.0e7` es $10^7 =$ `10000000`.

El siguiente ejemplo, es de gran relevancia para entender lo que significa la asignación de una variable.

Ejemplo 1.4 Escriba un programa que intercambie el contenido de dos variables dadas. Más precisamente, si se ingresan las variables:

```
a = 10;
b = -4;
```

y se realiza el programa de intercambio, se debe obtener como resultado: `a = -4` y `b = 10`.

Solución. Un error común es suponer que la solución es la siguiente:

```
a = b;
b = a;
```

donde se espera que el valor de `b` se copie en `a` mediante `a = b;` y luego que el valor de `a` se copie en `b` por medio de `b = a;`. Sin embargo, el problema es el siguiente: como `a = 10;` y `b = -4;`, al realizar la operación `a = b;` se copia el valor de `b` en `a`, por lo que ahora en `a` está almacenado el valor `-4`. Es decir, ambas variables contienen el mismo dato y con ello el valor `10` se perdió. Para apreciar esto, considere este proceso ilustrado en el Cuadro 1.1.

Cuadro 1.1 Descripción de las instrucciones del Ejemplo 1.4

Operación	**Resultado**
`a = 10; b = -4;`	`a` tiene a `10` y `b` tiene a `-4`
`a = b;`	`a` tiene a `-4` y `b` tiene a `-4`
`b = a;`	`a` tiene a `-4` y `b` tiene a `-4`

La manera correcta de intercambiar los valores entre dos variables, es crear una nueva variable, la cual guarde temporalmente el valor de una de ellas. Así, el programa de intercambio viene dado por:

```
tmp = a;
a = b;
b = tmp;
```

el cual se ilustra en el Cuadro 1.2.

Cuadro 1.2 Descripción de las instrucciones del Ejemplo 1.4

Operación	**Resultado**
`a = 10; b = -4;`	`a` tiene a `10` y `b` tiene a `-4`
`tmp = a;`	`a` tiene a `10`, `b` tiene a `-4` y `tmp` tiene a `10`
`a = b;`	`a` tiene a `-4`, `b` tiene a `-4` y `tmp` tiene a `10`
`b = tmp;`	`a` tiene a `-4` y `b` tiene a `10`

En general, para intercambiar los valores de dos variables es necesaria una variable adicional. En este ejemplo, esta fue nombrada `tmp`. □

Ejercicio 1.2 Escriba un programa que intercambie el contenido de cuatro variables: `a`, `b`, `c` y `d` en orden inverso. Por ejemplo, si `a = 10`, `b = -4`, `c = 19` y `d = 86`, entonces al intercambiar los valores se obtiene el resultado: `a = 86`, `b = 19`, `c = -4` y `d = 10`.

Observación: En general, para intercambiar los valores de una cantidad n de variables, seguirá siendo necesaria una (y solo una) variable auxiliar. Esto dado que el intercambio ocurre de dos en dos y no todas las variables a la vez.

1.3.1 Operaciones y expresiones

Al tener variables del tipo escalar ya definidas en el Workspace, estas pueden utilizarse para realizar las operaciones básicas descritas en el Cuadro 1.3.

Cuadro 1.3 Operaciones básicas

Símbolo	**Nombre de la operación**
`*`	Multiplicación
`^`	Potencia
`/`	División
`+`	Suma
`-`	Resta

Se debe tener en cuenta que MATLAB® respeta el orden usual de las operaciones combinadas y la prioridad de los paréntesis. Más aún, para este tipo de operaciones solo se deben utilizar los paréntesis redondos: `( )`, dado que los corchetes `[ ]` y las llaves `{ }` son usadas por MATLAB® para definir otros tipos de variables (no escalares).

Ejemplo 1.5 La fórmula

$$\sqrt{5} \; + \; \frac{3(5-10)^2 \; + \; 12(15+3)^{3-2}}{\left[(3-2) \; + \; (1-2)\right]^{\frac{5}{2}}}$$

se puede ingresar en MATLAB®, de la forma:

```
5^(1/2) + (3*(5-10)^2+12*(15+3)^(3-2)) / (((3-2)+(1-2))^(5/2))
```

Observación: Respecto al Ejemplo 1.5, es importante tener en cuenta que:

1. Aunque en la expresión original hay corchetes, al ingresar la fórmula en MATLAB®, estos se deben reemplazar por paréntesis redondos. Se debe tener en cuenta que los corchetes son utilizados por MATLAB® para construir o concatenar una matriz (véase Capítulo 2 para más detalles).

2. El uso de espacio entre operaciones y paréntesis no afecta la evaluación en la fórmula. Por tanto, el uso de espacios es útil para una mejor visualización por parte del usuario.

3. Ninguna operación es implícita en MATLAB®. Es decir, que debe colocarse mediante los símbolos anteriores. En particular, esto es relevante para la múltiplicación, la cual se suele omitir.

4. El uso de paréntesis es de suma importancia para indicarle a MATLAB® la forma adecuada en que se debe realizar un cálculo. Por ejemplo, si en el primer término del resultado, es decir: `5^(1/2)` equivalente a $5^{1/2} = \sqrt{5}$, se omiten los paréntesis: `5^1/2`, entonces se obtiene un resultado erróneo, dado que (en virtud de la prioridad de operaciones) `5^1/2` es equivalente a $\frac{5^1}{2} = 2.5 \neq \sqrt{5}$.

Ejercicio 1.3 Dado $x = 1$, se desea encontrar el valor de y definido por:

$$y = \left(\frac{x^2 - 3x + 2}{\frac{1}{2} + 5x} \right)^{2x+1} - \left[\frac{x^3 + \frac{1}{3}x^2 - \frac{5}{4}x + \pi}{\left(\frac{5}{6}\right)^2 (x-2) - \frac{1}{2x}} \right]^7$$

Escriba en MATLAB® un fragmento de código que lleve acabo dicha evaluación. Para ingresar el número irracional π, se puede hacer uso del comando: `pi`.

Observación: Si durante una operación aritmética se efectúa una división por cero, MATLAB® retorna la palabra reservada `Inf` (que significa *infinito*). Por otro lado, cuando una operación no está definida, por ejemplo `0/0`, se retorna `NaN` (que significa *no es un número*).

Otras operaciones predefinidas en MATLAB® son las siguientes:

1. `mod(a,b)`: consiste en el **módulo de la división** entre `a` y `b`, es decir, retorna el residuo que se obtiene al realizar la operación `a / b`. Por ejemplo, al escribir en la ventana de comandos las instrucciones:

   ```
   x = mod(10, 5)
   y = mod(-3, 2)
   z = mod(35, 6)
   ```

   se obtiene el resultado: `x = 0`, `y = 1` y `z = 5`.

2. `fix(x)`: se aplica a un número decimal `x` para eliminar sus decimales. Por ejemplo, al escribir

   ```
   x = fix(15.6)
   y = fix(10)
   z = fix(-7.5)
   ```

   se obtiene: `x = 15`, `y = 10` y `z = -7`.

3. `round(x,n)`: se aplica a un número decimal `x` para redondearlo a `n` decimales. Se debe tener en cuenta que `round(x,0)` es equivalente a escribir `round(x)`. Por ejemplo, al ejecutar:

```
x = round(15.6467)
y = round(10.3543137498, 4)
z = round(-7.5364532645, 2)
```

se obtiene: `x = 16`, `y = 10.3543` y `z = -7.5400`.

Ejemplo 1.6 Un relojero está trabajando en un cronómetro electrónico, el cual tiene un contador interno para la cantidad de segundos transcurridos. Solo cuenta los segundos. Ahora el relojero desea escribir un programa que dado el tiempo expresado en segundos, devuelva la cantidad de horas, minutos y segundos, en el formato usual de un cronómetro. Diseñe el programa que el relojero necesita.

Solución. Si `S` es la cantidad de segundos del cronómetro, considere el siguiente conjunto de instrucciones:

```
min = fix(S / 60);    % Candidad de minutos
seg = mod(S, 60);     % Cantidad de segundos
hor = fix(min / 60);  % Cantidad de horas
min = mod(min, 60);   % Cantidad de minutos
% Imprimir el tiempo en el formato hor:min:seg
[ num2str(hor) ':' num2str(min) ':' num2str(seg) ]
```

En particular, al ingresar `S = 35276`, se obtiene el resultado: `'9:47:56'`. Por otro lado, la última instrucción corresponde a una concatenación (véase Sección 2.2.2), donde se requiere el comando `num2str()` para convertir valores numéricos en cadena de caracteres. □

Ejercicio 1.4 Un cajero automático que cuenta con billetes de ¢1000, ¢2000, ¢5000, ¢10 000 y ¢20 000, despliega el monto a retirar procurando dar dicho monto con la mayor cantidad de billetes de alta denominación. Por ejemplo, si se retiran ¢2 373 000, el cajero deberá devolver: 118 billetes de ¢20 000, 1 de ¢10 000, 1 de ¢2000 y 1 de ¢1000. Escriba el programa de retiro para dicho cajero.

Acerca de los autores

Filánder Sequeira Chavarría

Nació el 19 de junio de 1986 en Guápiles, Limón. Obtuvo su bachillerato en Enseñanza de la Matemática asistida por Computadora, en el Instituto Tecnológico, Costa Rica, en el año 2007. Luego, en el 2010 obtuvo su maestría en Ciencias en Matemáticas Aplicadas, en la Universidad de Puerto Rico, Puerto Rico. Finalmente, recibió el título de Doctorado en Ciencias Aplicadas con mención en Ingeniería Matemática, en la Universidad de Concepción, Chile, en el año 2015. Desde el 2011 labora en la Escuela de Matemática de la Universidad Nacional, así como en la Escuela de Matemática de la Universidad de Costa Rica. En relación con la presente obra, él ha introducido numerosas veces el programa MATLAB® durante sus cursos y lo ha empleado en el desarrollo de sus investigaciones en el área de Matemática Aplicada, las cuales han dado origen a artículos científicos.

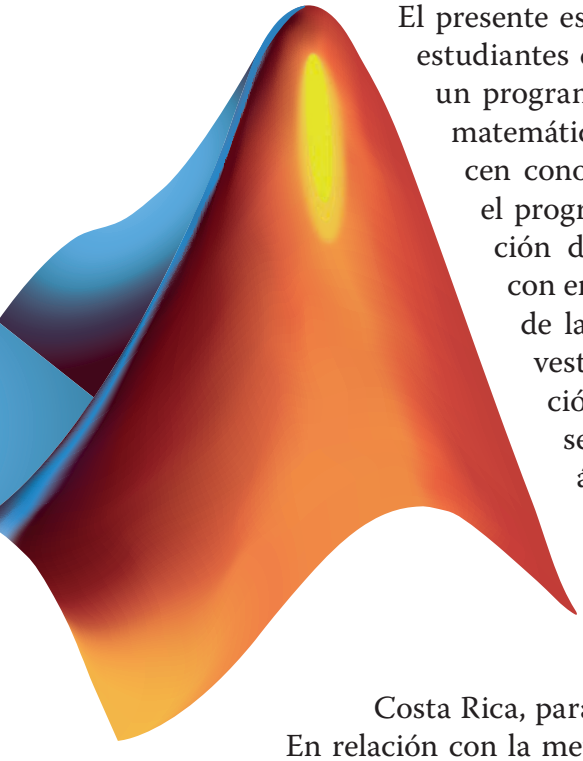
Esteban Segura Ugalde

Nació en San José, el 2 de octubre de 1984. Obtuvo su bachillerato en Matemática de la Universidad de Costa Rica en el año 2009. Luego, culminó su maestría en el año 2011 en la especialidad *Algorithmic, Symbolic Computation and Numerical Optimization*, en la Universidad de Limoges, Francia. Posteriormente, obtuvo en el año 2015 su doctorado en el Instituto XLIM de la Universidad de Limoges, en el tema *Structured Methods for Polynomial Computation*, con el cual obtuvo la mención Très Honorable. Actualmente es docente de la Escuela de Matemática e investigador del Centro de Investigación en Matemática Pura y Aplicada (CIMPA), de la Universidad de Costa Rica. Sus intereses se relacionan con el área del Álgebra Lineal Numérica donde, con respecto al presente trabajo, mantiene una continua labor docente y de investigación utilizando el programa MATLAB®.

Óscar Salas Huertas

Nació el 31 de diciembre de 1977 en San José, Costa Rica. Obtuvo su título de Licenciatura en Enseñanza de la Matemática en la Universidad Nacional, Costa Rica, en el 2004. En el 2003 se graduó de la maestría en Ciencias en Tecnología de los Medios, en la Universidad de Estudios de Pavia, Italia y posteriormente recibió el título de Doctorado en Matemática, también en la Universidad de Estudios de Pavia, Italia, en el 2008. Desde ese año labora en la Escuela de Matemática de la Universidad Nacional así como en la Escuela de Matemática de la Universidad de Costa Rica. En relación con la presente obra, el autor desde su formación doctoral ha utilizado el programa MATLAB® en el desarrollo de sus investigaciones en el área de Matemática Aplicada y, actualmente, este software representa una herramienta fundamental en sus labores de docencia universitaria.

Introducción a MATLAB



El presente escrito tiene como principal objetivo apoyar a los estudiantes durante sus primeros pasos en el aprendizaje de un programa computacional especializado en el área de la matemática y aplicaciones. Más precisamente, se introducen conocimientos básicos que se deben dominar sobre el programa MATLAB, el cual es idóneo para la ejecución de cálculos numéricos que involucran matrices con entradas reales o complejas, convirtiéndolo en una de las herramientas más utilizadas para realizar investigaciones afines a la disciplina sobre computación científica. Aunque el presente material puede ser empleado por cualquier lector interesado en el área, es importante aclarar que este está dirigido a estudiantes de pregrado, en especial a aquellos en carreras afines en informática y matemática. Un ejemplo de ello es la asignatura MA1006: Introducción al Análisis Numérico de la carrera de Ingeniería Civil de la Universidad de Costa Rica, para la cual originalmente se confeccionó esta obra. En relación con la metodología que busca el documento, esta procura dotar al lector de las instrucciones y comandos que ofrece MATLAB de una manera resumida y clara, brindando además diversos ejemplos representativos, que ilustran posibles aplicaciones de estos.

EDITORIAL UCR