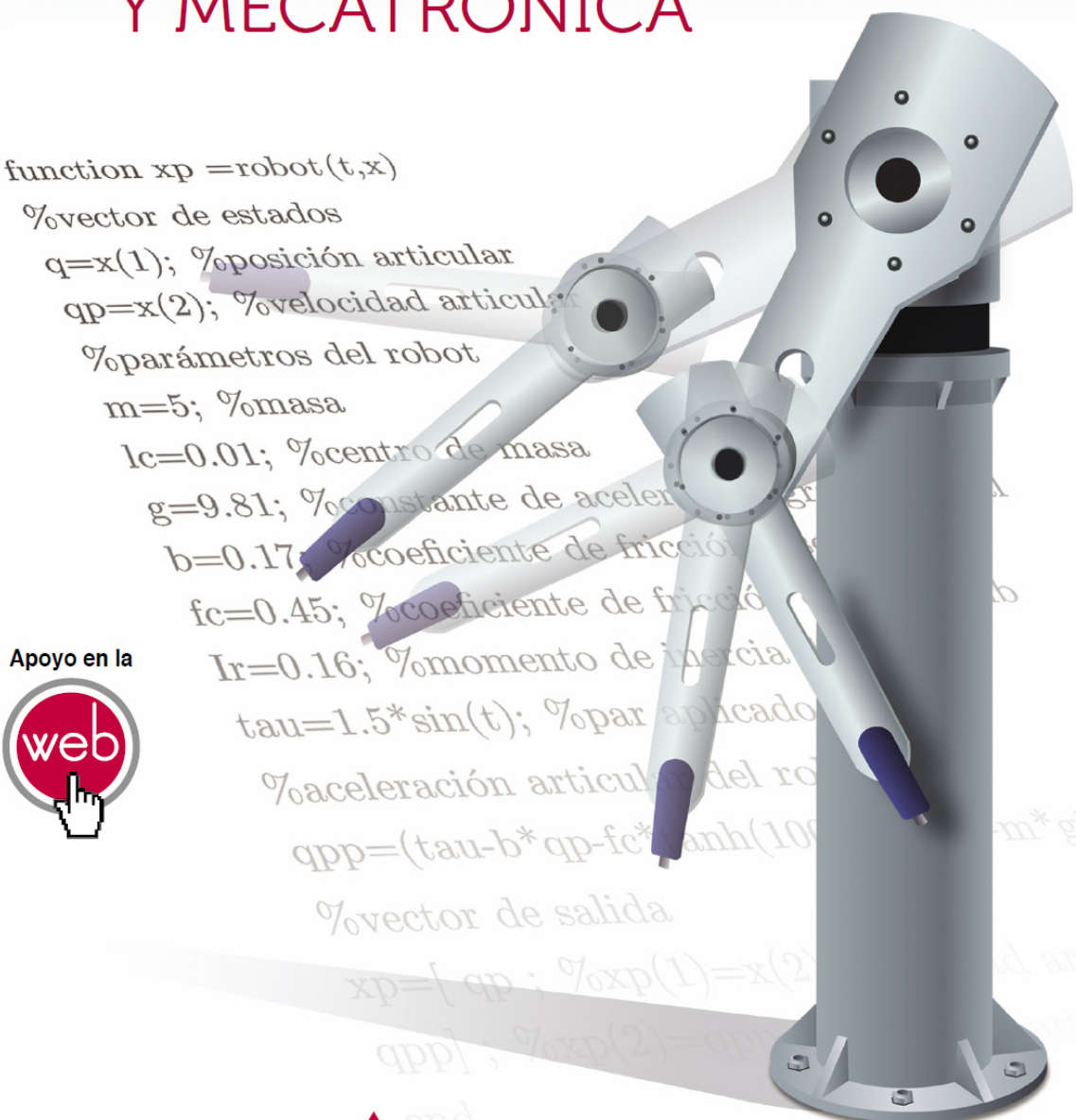
FERNANDO REYES CORTÉS

MATLAB

APLICADO A ROBÓTICA Y MECATRÓNICA



Apoyo en la web

Alfaomega

MATLAB

APLICADO A ROBÓTICA Y MECATRÓNICA

FERNANDO REYES CORTÉS

Alfaomega

Editor
Francisco Javier Rodríguez Cruz
jrodriguez@alfaomega.com.mx

Director Editorial
Marcelo Grillo Giannetto
mgrillo@alfaomega.com.mx

Datos catalográficos
Reyes Cortés, Fernando
MATLAB aplicado a Robótica y Mecatrónica
Primera Edición
Alfaomega Grupo Editor, S.A. de C.V., México
ISBN: 978-607-707-357-4
Formato: 17 x 23 cm Páginas: 460

MATLAB aplicado a Robótica y Mecatrónica
Fernando Reyes Cortés

Primera edición: Alfaomega Grupo Editor, México, enero 2012

Pitágoras 1139, Col. Del Valle, 03100, México D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-707-357-4

Nota importante:
La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos, han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. ALFAOMEGA GRUPO EDITOR, S.A. de C.V. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele.

Impreso en México. Printed in Mexico.

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Pitágoras 1139, Col. Del Valle, México, D.F. – C.P. 03100.
Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Carrera 15 No. 64 A 29, Bogotá, Colombia,
Tel.: (57-1) 2100122 – Fax: (57-1) 6068648 – E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – Dr. La Sierra 1437, Providencia, Santiago, Chile
Tel.: (56-2) 235-4248 – Fax: (56-2) 235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. – Paraguay 1307 P.B. Of. 11, C.P. 1057, Buenos Aires, Argentina, – Tel./Fax: (54-11) 4811-0887 y 4811 7183 – E-mail: ventas@alfaomegaeditor.com.ar

Acerca del autor

Dr. José Fernando Reyes Cortés. Es profesor investigador titular C de la Facultad de Ciencias de la Electrónica, Universidad Autónoma de Puebla. En 1984 obtuvo la Licenciatura en Ciencias de la Electrónica en la Facultad de Ciencias Físico Matemáticas de la Universidad Autónoma de Puebla. En 1990 obtuvo la Maestría en Ciencias con Especialidad en Electrónica en el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE). Bajo la asesoría del Dr. Rafael Kelly, en 1997 culminó el Doctorado en Ciencias con Especialidad en Electrónica y Telecomunicaciones en el Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE).

Es miembro del Sistema Nacional de Investigadores desde 1993 a la fecha. Actualmente es nivel I. Es autor de más de 150 artículos nacionales e internacionales. Ha dirigido 30 proyectos científicos. Ha graduado a más de 100 tesistas de los niveles de ingeniería, maestría y doctorado.

Es autor del libro **Robótica. Control de robots manipuladores** editado por Alfaomega donde se presenta la dinámica de robots manipuladores y la técnica moderna de moldeo de energía para diseñar nuevos algoritmos de control.

Como catedrático ha impartido más de 50 cursos del área de control y robótica a nivel licenciatura y posgrado. Fundador del Laboratorio de Robótica y Control de la Facultad de Ciencias de la Electrónica donde ha puesto a punto 30 prototipos de mecatrónica y robótica. Es Premio Estatal de Tecnologías y Ciencias de la Ingeniería en noviembre 2000 por el Gobierno del Estado de Puebla. Premio en Ingeniería y Tecnología de la Ciudad de Puebla en abril 2010.

Ha sido Secretario de Investigación y Estudios de Posgrado de la Facultad de Ciencias de la Electrónica desde febrero del 2002 a la fecha. Fundador y responsable del Cuerpo Académico de Robótica y Control en 2001. Fue responsable de la creación de la Ingeniería Mecatrónica, y de los Posgrado de Ciencias de la Electrónica e Ingeniería Electrónica e ingreso de ambos posgrados al PNPC de CONACyT.

A la persona que me dio la vida: Alicia Cortés Castillo; por su apoyo y paciencia a mi esposa Silvia y mis tesoros Luis Fernando y Leonardo. Por la formación transmitida de Angela Castillo Merchant, Luis Manuel Cortés Castillo, Chela, Lety y Jorge.

Por la ayuda brindada de la maestra Viky.

Mensaje del Editor

Una de las convicciones fundamentales de Alfaomega es que los conocimientos son esenciales en el desempeño profesional, ya que sin ellos es imposible adquirir las habilidades para competir laboralmente. El avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos, y de acuerdo con esto Alfaomega publica obras actualizadas, con alto rigor científico y técnico, y escritas por los especialistas del área respectiva más destacados.

Conciente del alto nivel competitivo que debe de adquirir el estudiante durante su formación profesional, Alfaomega aporta un fondo editorial que se destaca por sus lineamientos pedagógicos que coadyuvan a desarrollar las competencias requeridas en cada profesión específica.

De acuerdo con esta misión, con el fin de facilitar la comprensión y apropiación del contenido de esta obra, cada capítulo inicia con el planteamiento de los objetivos del mismo y con una introducción en la que se plantean los antecedentes y una descripción de la estructura lógica de los temas expuestos, asimismo a lo largo de la exposición se presentan ejemplos desarrollados con todo detalle y cada capítulo concluye con un resumen y una serie de ejercicios propuestos.

Además de la estructura pedagógica con que están diseñados nuestros libros, Alfaomega hace uso de los medios impresos tradicionales en combinación con las Tecnologías de la Información y las Comunicaciones (TIC) para facilitar el aprendizaje. Correspondiente a este concepto de edición, todas nuestras obras de la Serie Profesional a la que pertenece este título tienen su complemento en una página Web en donde el alumno y el profesor encontrarán lecturas complementarias, código fuente de los programas desarrollados así como la solución y la respuesta de los problemas propuestos.

Los libros de Alfaomega están diseñados para ser utilizados en los procesos de enseñanza-aprendizaje, y pueden ser usados como textos en diversos cursos o como apoyo para reforzar el desarrollo profesional, de esta forma Alfaomega espera contribuir así a la formación y al desarrollo de profesionales exitosos para beneficio de la sociedad y del mundo del conocimiento.

Contenido

Plataforma de contenidos interactivos

Para tener acceso al código fuente de los programas de ejemplos y ejercicios presentados en **MATLAB Aplicado a Robótica y Mecatrónica**, siga los siguientes pasos:

1) Ir a la página

`http://virtual.alfaomega.com.mx`

2) Registrarse como usuario del sitio.

3) En el catálogo identificar este libro y descargar el material adicional.

Simbología e iconografía utilizada

Diversos recursos didácticos están presentes en esta obra; particularmente se resaltan las siguientes herramientas:

Librerías

La sintaxis de las librerías que se desarrollan son representadas como:

if $\dot{x}$=robot(t, x)

Los ejemplos ilustrativos se presentan de la siguiente manera:

♣ ♣ Ejemplo 5.1

El enunciado de cada ejemplo se encuentra dentro de un recuadro con fondo gris. Los ejemplos son presentados por nivel de complejidad, para el nivel simple o básico se emplea la marca ♣, ejemplos del nivel intermedio ♣♣ y complejos son denotados por ♣♣♣

Solución

Se detalla la respuesta de cada ejemplo por ecuaciones y programas en código fuente. Adicionalmente, todos los ejemplos incluyen un número de referencia que identifica al capítulo donde fue definido.

Todos los programas de esta obra han sido implementados en lenguaje **MATLAB** versión 11. El código fuente se identifica por el siguiente recuadro:

Código Fuente 5.1 robot.m

```
%MATLAB Aplicado a Robótica y Mecatrónica.
%Editorial Alfaomega,  Fernando Reyes Cortés.
%Simulación de robots manipuladores
```

robot.m

```
1  clc;
2  clear all;
3  close all;
4  format short g
5  ti=0; h=0.001; tf = 10;%vector tiempo
6  t=ti:h:tf;%tiempo de simulación
7  ci=[0; 0; 0; 0];
8  opciones=odeset('RelTol',1e-3,'InitialStep',1e-3,'MaxStep',1e-3);
9  [t,x]=ode45('robot',t,ci,opciones);
10 plot(x(:,1),x(:,2))
```

Para representar una idea general de una instrucción de programación, se emplea pseudo-código como el siguiente:

Estructura de código 5.1

Pseudo código

```
while k<1000
    qp(k)=(q(k)-q(k-1))/h;
    if q(k)>100
        for j=1:1000
            qpp(j)=robot(q(j),qp(j));
        end
    end
    k=k+1;
    otro grupo de instrucciones;
end
```

Prólogo

Robótica y mecatrónica representan en la actualidad áreas estratégicas y claves para todo país que aspire a la modernidad y bienestar, ya que su impacto no sólo repercute en aspectos políticos y económicos, también forma parte importante de la vida cotidiana, educación, cultura, y en todos los ámbitos de la sociedad.

La simulación es una herramienta imprescindible para reproducir los fenómenos físicos de un robot o de un sistema mecatrónico, permite estudiar y analizar a detalle los aspectos prácticos que intervienen en tareas específicas que debe realizar un robot industrial. La simulación es un proceso previo a la etapa experimental donde es posible entender los conceptos claves de la robótica y mecatrónica. Bajo este escenario se ubica la presente obra a través de la propuesta de un conjunto de librerías para **MATLAB** que permitan realizar estudio, análisis, simulación y aplicaciones de robots manipuladores y sistemas mecatrónicos.

Esta obra presenta la propuesta y desarrollo de una clase particular de librearías *toolbox* para robótica y mecatrónica. Las librerías son funciones en código fuente para **MATLAB** que permiten modelar la cinemática directa e inversa, transformaciones homogéneas (rotación y traslación), dinámica, identificación paramétrica, control de robots manipuladores. El contenido y material incluido en el libro está dirigido a estudiantes de ingeniería en electrónica, sistemas, computación, industrial, robótica y mecatrónica. No obstante, también puede ser adecuado para nivel posgrado.

La organización de este libro consta de cuatro partes: la **Parte I Programación** incluye dos capítulos. El **Capítulo 1 Conceptos básicos** de programación del lenguaje de **MATLAB** tiene la finalidad de que el lector adquiera los conocimientos necesarios para adquirir solvencia en programar aplicaciones en código fuente. El capítulo 2 **Métodos numéricos** presenta aspectos relacionados con las técnicas de métodos numéricos enfocados a resolver problemas de diferenciación e integración numérica de sistemas dinámicos.

La **Parte II Cinemática** está relacionada con el análisis cinemático de las principales clases de robots manipuladores. Se componen de tres capítulos; el

capítulo 3 **Preliminares matemáticos** contiene las bases de las reglas de las matrices de rotación, traslación y transformaciones homogéneas. En el capítulo 4 **Cinemática directa** se detalla la metodología de Denavit-Hartenberg para describir la cinemática directa; asimismo se describe la cinemática inversa, cinemática diferencial y jacobianos. Un compendio de librerías se desarrollaron con variables simbólicas y aplicaciones numéricas para los principales tipos de robots manipuladores en el capítulo 5 **Cinemática directa cartesiana**.

La **Parte III Dinámica** describe la técnica de simulación para sistemas dinámicos e identificación paramétrica. Formada por el capítulo 6 **Dinámica** donde se realiza simulación de sistemas mecatrónicos y robots manipuladores con la estructura de una ecuación diferencial ordinaria de primer orden descrita por variables de estado. El capítulo 7 **Identificación paramétrica** presenta cinco esquemas de regresión con la técnica de mínimos cuadrados. Se proponen extensos ejemplos para ilustrar el procedimiento para encontrar el valor numérico de los parámetros del robot.

Finalmente, la **Parte IV** capítulo 8 **Control de posición** contiene la simulación de algoritmos clásicos como el control PD y PID; así como simulación de nuevas estrategias de control con mejor desempeño que pertenecen a la filosofía de diseño moderno conocida como *moldeo de energía*. Aplicaciones de control punto a punto se ilustran para describir la técnica de simulación.

La realización de esta obra no hubiera sido posible sin el apoyo de la ***Benemérita Universidad Autónoma de Puebla*** (BUAP), particularmente agradezco a la Facultad de Ciencias de la Electrónica por todas las facilidades brindadas. El autor desea agradecer principalmente al Dr. Jaime Cid, M. C. Fernando Porras, Dra. Aurora Vargas, Dr. Sergio Vergara, Dra. Amparo Palomino y al M. C. Pablo Sánchez, así como al programa de financiamiento de proyectos de investigación de la Vicerrectoría de Investigación y Estudios de Posgrado (VIEP), especialmente a la Dra. Rosa Graciela Montes y al Dr. Pedro Hugo Hernández Tejeda.

Fernando Reyes Cortés
H. Puebla de Z., a 3 de diciembre de 2011
Facultad de Ciencias de la Electrónica
Benemérita Universidad Autónoma de Puebla

Parte I

Programación

La **Parte I** de la presente obra está destinada a estudiar el lenguaje de programación de **MATLAB** y la implementación de métodos numéricos aplicados a la robótica y mecatrónica. Esta primera parte se compone de dos capítulos. El capítulo 1 **Conceptos básicos** presenta los aspectos básicos del lenguaje **MATLAB** y tiene la finalidad de que el lector adquiera los conocimientos fundamentales para adquirir solvencia en programar en código fuente. El capítulo 2 **Métodos numéricos** presenta aspectos relacionados con las técnicas de métodos numéricos enfocados a resolver problemas de modelado y control de las áreas de robótica y mecatrónica.

Capítulo 1 Conceptos básicos

Descripción de los principales componentes del ambiente de programación de **MATLAB**. Sintaxis y programación, declaración de variables, tipo de datos, operadores, programación, manipulación numérica de matrices y arreglos, funciones, gráficas, funciones importantes, matrices, operadores, instrucciones, lazos de programación, etcétera. Un conjunto de ejemplos se desarrollan para ilustrar los temas presentados.

Capítulo 2 Métodos numéricos

Conocer los principales métodos de diferenciación e integración numérica que se aplican en robótica y mecatrónica. Sistemas de ecuaciones lineales, método de Euler e integración numérica (trapezoidal, Simpson). Simulación de sistemas dinámicos con la técnica de Runge-Kutta.

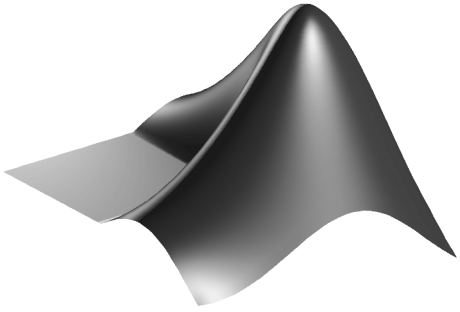
La parte I concluye con referencias selectas y un conjunto de problemas para mejorar las habilidades y conocimientos de los capítulos tratados.

Referencias selectas

Problemas propuestos

Capítulo 1

Conceptos básicos



Objetivos

Presentar la plataforma y ambiente de programación de **MATLAB** con los fundamentos y bases de su lenguaje enfocado para adquirir solvencia y dominio en la implementación de algoritmos con aplicaciones en mecatrónica y robótica.

Objetivos particulares:

Aprender el Lenguaje **MATLAB**.

Conocer los operadores.

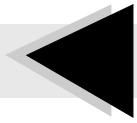
Programar con matrices y arreglos.

Manejar funciones e instrucciones.

Graficar funciones y datos en 2D y 3D.

Desarrollar formatos para manejar datos experimentales.

1.1 Introducción



Hoy en día, **MATLAB** es un lenguaje de programación matemático de alto nivel integrado con entorno gráfico amigable, visualización de datos, funciones, gráficas 2D y 3D, procesamiento de imágenes, video, computación numérica para desarrollar algoritmos matemáticos con aplicaciones en ingeniería y ciencias exactas. Particularmente, en ingeniería es una herramienta muy poderosa para realizar aplicaciones en mecatrónica, robótica, control y automatización.

MATLAB es un acrónimo que proviene de **matrix laboratory** (*laboratorio matricial*) creado por el profesor y matemático Cleve Moler en 1970. La primera versión de **MATLAB** fue escrita en lenguaje fortran la cual contempló la idea de emplear subrutinas para los cursos de álgebra lineal y análisis numérico de los paquetes LINPACK y EISPACK; posteriormente se desarrolló software de matrices para acceder a esos paquetes sin necesidad de usar programas en fortran. La empresa The Mathworks (`http://www.mathworks.com`), propietaria de **MATLAB** fue fundada por Jack Little y Cleve Moler en 1984. El actual paquete de **MATLAB** se encuentra escrito en lenguaje C, y su primera versión en este lenguaje fue escrita por Steve Bangert quien desarrolló el intérprete parser. Steve Kleiman implementó los gráficos, las rutinas de análisis, la guía de usuario, mientras que la mayoría de los `archivos.m` fueron escritos por Jack Little y Cleve Moler. Actualmente, el lenguaje de programación de **MATLAB** proporciona un sencillo acceso a algoritmos numéricos que incluyen matrices.

Las versiones recientes del lenguaje **MATLAB** se caracterizan por ser multiplataforma, es decir se encuentra disponible para sistemas operativos como Unix, Windows y Apple Mac OS X. **MATLAB** posee varias características computacionales y visuales, entre las que sobresalen la caja de herramientas (*toolbox*) la cual representa un amplio compendio de funciones y utilerías para analizar y desarrollar una amplia gama de aplicaciones en las áreas de ingeniería y ciencias exactas. Un rasgo distintivo de **MATLAB** es que ofrece un entorno gráfico de programación amigable al usuario a través de herramientas y utilerías para realizar simulación de sistemas dinámicos, análisis de datos, procesamiento de imágenes y video, gráficas y métodos

de visualización, desarrollo de interfaces de usuario (GUI); también permite realizar interfaces para sistemas electrónicos, por ejemplo adquisición de datos con una versatilidad de tarjetas de instrumentación electrónica comerciales con plataforma de microprocesadores, DSP's, PIC's, FPGA's, manejo de puertos como USB, COM, paralelo y diseños electrónicos propios.

MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones: plataforma de simulación multidominio ***Simulink*** y ***GUIDE*** editor de interfaces de usuario. Se recomienda al lector visitar:

```
www.mathworks.com/products/matlab
```

en ese sitio WEB se pueden encontrar productos actuales de **MATLAB**, manuales, notas técnicas, cursos, aplicaciones y foros internacionales, así como sociedades de desarrollo. Actualmente, **MATLAB** se ha convertido en un referente de cómputo a nivel internacional, debido que en la mayoría de las universidades, centros de investigación e industria lo utilizan para desarrollar y llevar a cabo proyectos de ciencia y tecnología.

Específicamente en el área de ingeniería **MATLAB** permite realizar simulaciones de sistemas mecatrónicos y robots manipuladores. El proceso de simulación resulta importante cuando no se dispone de una adecuada infraestructura experimental. Sin embargo, la simulación depende de un buen modelo matemático que permita reproducir fielmente todos los fenómenos físicos del sistema real. La simulación es flexible ya que permite detectar posibles deficiencias en el diseño del modelado. Una vez depurado el modelo dinámico, entonces la simulación facilita el proceso para analizar, estudiar y comprender el comportamiento de la dinámica del sistema. Esta etapa es fundamental para el diseño de algoritmos de control. Cuando el modelo matemático del sistema mecatrónico o robot manipulador es lo suficientemente completo entonces la simulación proporciona un medio virtual del sistema real.

El propósito de este capítulo se ubica en preparar al lector para conocer mejor el entorno de programación, programar y manejar en forma solvente el lenguaje de **MATLAB**. En otras palabras, proporcionar todos los elementos necesarios de programación (variables, datos, instrucciones y funciones) para realizar diversas

aplicaciones para las áreas de robótica y mecatrónica.

1.2 Componentes

El ambiente de programación de **MATLAB** es amigable al usuario, y está compuesto por una interface gráfica con varias herramientas distribuidas en ventanas que permiten programar, revisar, analizar, registrar datos, utilizar funciones, historial de comandos y desarrollar diversas aplicaciones. La forma de iniciar el paquete **MATLAB** es realizando doble *click* sobre el icono colocado en el escritorio de Windows (*Windows desktop*). La pantalla principal de **MATLAB** se presenta en la figura 1.1 la cual contiene la interface gráfica de usuario con varias ventanas como la de comandos (*Command Window* $\boldsymbol{fx}$ >>), manejo de archivos, espacio de trabajo (*Workspace*), historial de comandos (*Command History*), directorio de trabajo y aplicaciones como Simulink.

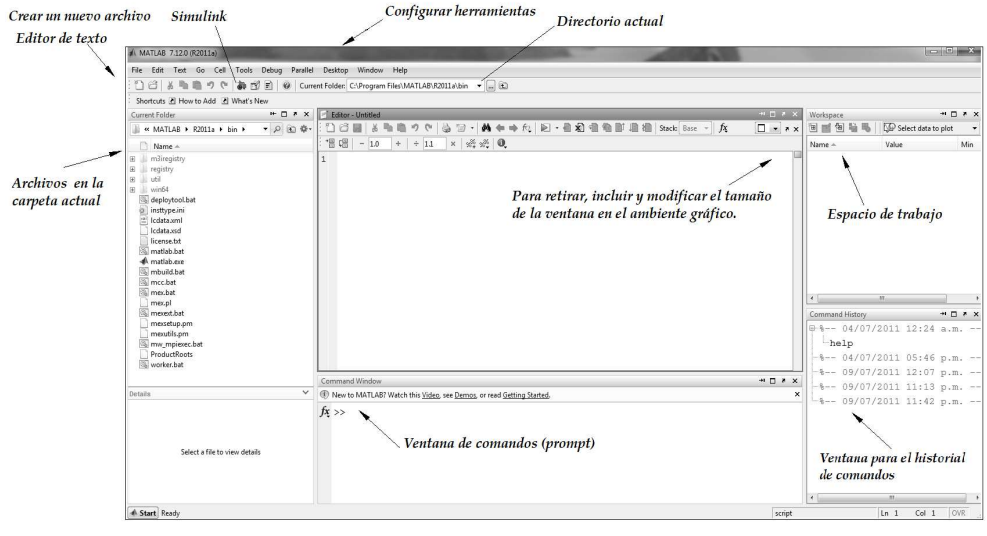


Figura 1.1 Ambiente de programación de **MATLAB**.

Es recomendable personalizar el ambiente de programación de **MATLAB** para una fácil interacción con manejos de archivos, datos y herramientas de programación. Por ejemplo, para integrar la ventana del editor dentro del ambiente gráfico se logra oprimiendo el icono del editor de texto colocado en la esquina superior izquierda,

justo abajo de la opción `File` del menú principal, posteriormente se hace *click* sobre la flecha colocada en la esquina superior derecha de la ventana del editor. La figura 1.2 muestra el ambiente configurado.

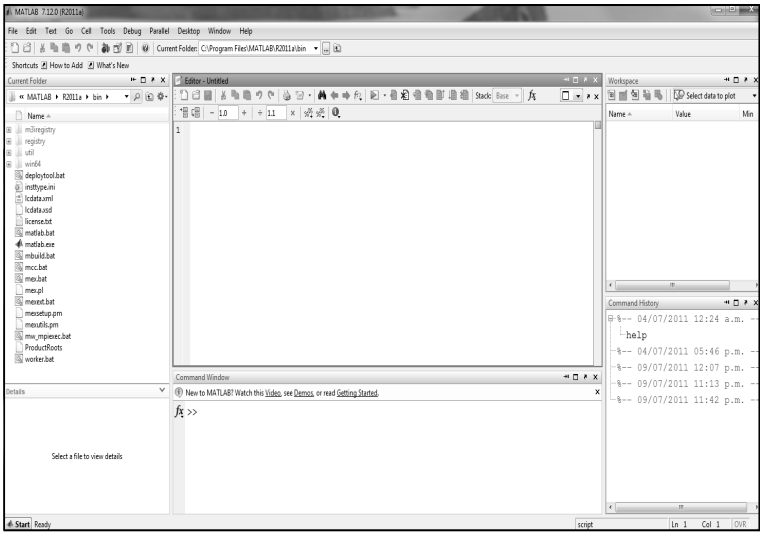


Figura 1.2 Editor de texto integrado en el ambiente de programación.

Generalmente el usuario registra sus archivos en una carpeta o directorio predefinido; para dar de alta dicho directorio al momento de simular aparecerá el mensaje que se muestra en la figura 1.3. Seleccionar `Add to Path` para que **MATLAB** realice la simulación desde esa trayectoria de trabajo, de esta forma las trayectorias de otros directorios que estén habilitadas no se perderán, por lo que la trayectoria del usuario se añade a las ya existentes.
Las componentes principales del ambiente de programación de **MATLAB** son:

- Herramientas de escritorio y ambiente de desarrollo.
- Librerías.
- Lenguaje **MATLAB**.
- Gráficas.
- Interfaces externas/API.

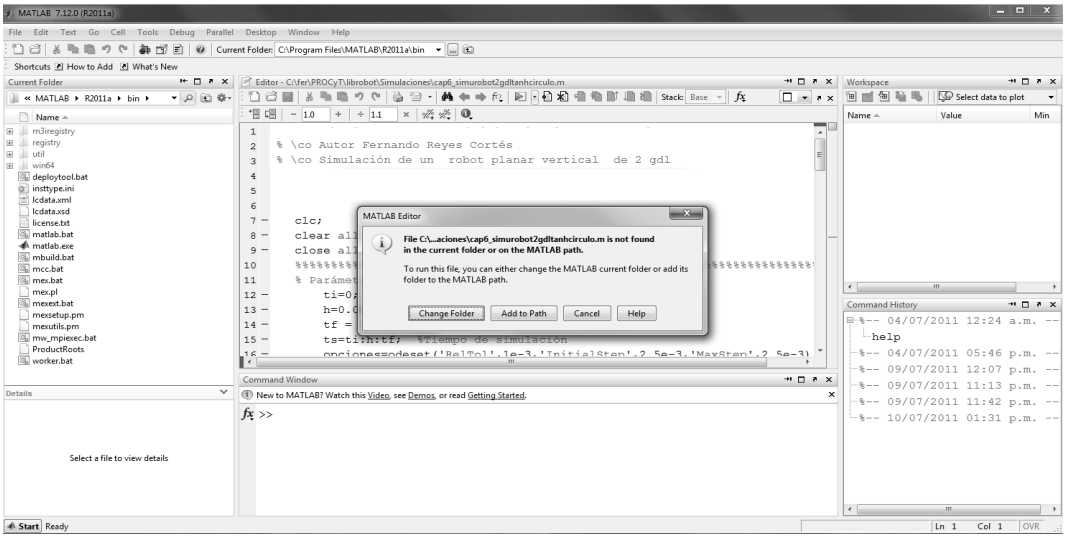


Figura 1.3 Habilitación del directorio de trabajo de usuario.

1.2.1 Herramientas de escritorio y ambiente de desarrollo

MATLAB tiene un conjunto de herramientas que facilitan el uso de funciones, comandos, programas, datos, variables y archivos (*desktop tools and development environment*). La mayoría de las herramientas son interfaces gráficas donde el usuario interacciona con comandos, funciones, editor de texto, variables y datos. Por ejemplo **MATLAB** *desktop*, *Command Windows*, Simulink, ayuda en línea, analizador de código, navegador de archivos, etcétera.

1.2.2 Librerías

Las librerías de **MATLAB** son un conjunto muy grande de funciones y algoritmos matemáticos que van desde funciones trigonométricas, operaciones básicas con matrices hasta funciones complejas como análisis de regresión, inversa de matrices, funciones Bessel, transformada rápida de Fourier, Laplace, programación de DSP's, microprocesadores, procesamiento de imágenes, video y tarjetas de instrumentación electrónica, entre otras ventajas.

1.2.3 Lenguaje

El lenguaje de programación de **MATLAB** es de alto nivel y permite programar matrices, arreglos e incorporar instrucciones de control de flujo del programa, funciones, comandos y estructura de datos.

MATLAB contiene una diversidad en comandos que facilitan al usuario la implementación del problema a simular. Los comandos son funciones muy específicas que tienen un código depurado y que no se encuentra disponible al usuario.

El lenguaje de programación de **MATLAB** contiene todos los elementos de programación necesarios para desarrollar aplicaciones en realidad virtual, programación de robots manipuladores, análisis de regresión y estadísticos de señales experimentales, procesamiento y extracción de rasgos distintivos de imágenes y video, así como varias aplicaciones más en ingeniería y ciencias exactas.

1.2.4 Gráficos

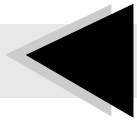
MATLAB contiene un enorme número de funciones que facilitan la representación gráfica y visualización de variables, funciones, vectores, matrices y datos que pueden ser graficados en 2 y 3 dimensiones. Incluye también funciones de alto nivel para el análisis y procesamiento de gráficas, datos experimentales o de simulación, video e imágenes, animación y presentación de sólidos, así como el desarrollo de aplicaciones con interface gráfica para incluir menús con botones, barras deslizadoras, diagramas a bloques, instrumentos de medición y ventanas para seleccionar opciones o herramientas de la aplicación.

1.2.5 Interfaces externas/API

Hay varias herramientas especiales para escribir programas en lenguaje C, C++, fortran y Java que interactúen con programas en lenguaje **MATLAB** facilitando el proceso de llamada a rutinas desde **MATLAB** y pase de parámetros, así como la

programación de puertos (USB, COM, paralelo y serial) que permiten conectar tarjetas de instrumentación electrónica para adquisición de datos y evaluación experimental de algoritmos de control.

1.3 Inicio



Una vez instalado **MATLAB**, la forma más simple de interaccionar con este paquete es introduciendo expresiones directamente en la venta de comandos, por ejemplo iniciando con el tradicional letrero "Hola mundo..." sobre el *prompt*:

```
fx >>  disp('Hola mundo...')  ↩

           Hola mundo...
fx >>  9+9 oprimir la tecla Enter  ↩
       ans=
```

18

```
fx >>  sin(10) oprimir Enter  ↩
       ans=
```

−0.5440

Cuando no se especifique el nombre de la variable, **MATLAB** usará el nombre por *default* `ans` que corresponde a un nombre corto de *answer*. Al especificar una variable el resultado se desplegará con el nombre de esa variable, por ejemplo:

```
fx >>  z=9+9 Enter  ↩
       z=
```

18

```
fx >>  y=sin(10) Enter  ↩
       y=
```

−0.5440

MATLAB funciona como calculadora, sobre el `prompt` de la ventana de comandos se pueden escribir expresiones aritméticas mediante los operadores "+","-","/", "*", "^" para la suma, resta, división, multiplicación y potencia, respectivamente. La tabla 1.1 contiene los operadores aritméticos básicos.

Por ejemplo, similar a una calculadora se pueden evaluar expresiones como:

```
fx >>  (10.3+8*5/3.33)^5  ↩
       ans=
```

$$5.5296e+006$$

```
fx >>  ans^2  ↩
       ans=
```

$$3.0576e+013$$

Tabla 1.1 `Operadores aritméticos básicos`

Adición	+
Sustracción	-
Multiplicación	*
División	/
División left	\
Potencia	^
Evaluación	( )

MATLAB utiliza el operador **;** para deshabilitar la opción de desplegado en la ventana de comandos. Por ejemplo,

```
fx >>  z=9+9;  ↩
fx >>
```

es decir, con el operador ; **MATLAB** no exhibirá ningún valor de variable o función.

El lenguaje **MATLAB** permite insertar comentarios usando el operador % el cual deberá emplearse por cada línea de comentarios. Por ejemplo:

```
fx >> w=10+20%Suma de dos enteros. ↩
      w=
```

30

```
fx >> sign(-8999.4)%Obtiene el signo de un número. ↩
      ans=
```

-1

Obsérvese que al ejecutarse la instrucción o la sentencia el comentario no se despliega. Los comentarios sirven para documentar aspectos técnicos de un programa.

Herramienta de ayuda de MATLAB

MATLAB tiene una extensa gama de herramientas para solicitar ayuda en línea (*help*), ya sea de manera general o en forma específica buscar información para una instrucción comando o función, así como sus principales características y ejemplos didácticos (*demos*).

En la ventana de comandos puede solicitar ayuda de la siguiente forma:

```
fx >> help ↩
fx >> helpwin ↩
fx >> help general ↩
fx >> helpdesk ↩
fx >> help if ↩
fx >> help demo ↩
```

La ayuda en línea del programa **MATLAB** también proporciona información sobre los `demos` y tutoriales que facilitan el aprendizaje del paquete.

Para suspender la ejecución de un programa

Cuando **MATLAB** se encuentra realizando la ejecución de un programa o cálculo computacional en la esquina inferior izquierda aparecerá un icono que indica `busy`. Sin embargo, hay ocasiones que toma mucho tiempo la ejecución del programa debido a que se encuentra realizando operaciones complicadas, funciones discontinuas o con números muy grandes, en cuyo caso despliega `Inf`, que significa infinito.

```
fx >> sinh(1345e34)%Obtiene el seno hiperbólico de un número. ↩
      ans=

                              Inf
```

La siguiente expresión genera un lazo infinito de ejecución. Es decir el programa se quedará de manera indefinida en ejecución.

```
fx >> while 1 end%Genera un ciclo infinito de ejecución del programa.
↩
```

En este caso, hay ocasiones donde es recomendable interrumpir el proceso mediante la combinación de teclas `CRTL C` (presionar simultáneamente `CONTROL` y la tecla `C`).

> Para que tenga efecto `CRTL C` es muy importante que se encuentre sobre la ventana de comandos; si no es el caso primero haga *click* sobre dicha ventana, y posteriormente `CRTL C`. De lo contrario no tendrá efecto la acción de suspender programa.

Finalizar la sesión de MATLAB

Para finalizar **MATLAB** simplemente escribir en la ventana de comandos lo siguiente:

```
fx >> quit ↩
fx >> exit ↩
```

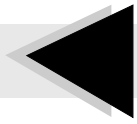
También se pueden usar las siguientes opciones para finalizar la sesión con **MATLAB**:

Desde el teclado, presionar simultáneamente `CRTL Q` o `CONTROL` con `q`

Presionar simultáneamente las teclas `ALT F` y después la opción `Exit`

Con el *mouse* sobre el menú de **MATLAB** *click* sobre `File` y *click* en `Exit`.

1.4 Lenguaje



El lenguaje de programación de **MATLAB** está compuesto por un conjunto de reglas gramaticales y sintaxis para escribir correctamente las variables, constantes, operadores, expresiones, funciones y todos los elementos que forman parte de la programación.

Las variables, constantes, operadores y funciones forman una **expresión** la cual será procesada por un analizador léxico y sintáctico antes de ser ejecutada por la computadora. A diferencia de otros lenguajes, las expresiones en **MATLAB** involucran matrices.

1.4.1 Variables

En el lenguaje de **MATLAB** las variables no requieren ningún tipo de declaración o definición. Esto tiene varias ventajas ya que facilita la programación. Cuando **MATLAB** encuentra el nombre de una nueva variable, automáticamente crea la variable y le asigna una localidad de memoria. Por ejemplo,

```
fx >> error_posición=20
```

esta variable es vista por **MATLAB** desde el punto de vista de programación como una matriz de dimensión 1×1.

Los nombres usados para referenciar a las variables, funciones y otro tipo de objetos o estructuras definidos por el usuario se les conocen como **identificadores**. Los nombres de identificadores constan de una letra del idioma castellano como caracter

inicial (**con excepción de la letra ñ y sin acentos**) seguido de cualquier número de letras, dígitos, y también se puede usar el guión bajo (*underscore* _). MATLAB distingue las letras mayúsculas de las minúsculas; por ejemplo la variable `A` es diferente a la variable `a`. La longitud de los nombres de las variables puede ser cualquier número de caracteres. Sin embargo, MATLAB usa los primeros 63 caracteres del nombre (*namelengthmax*=63) e ignora los restantes. Por lo tanto, para distinguir variables es importante escribir los nombres de las variables con un máximo de 63 caracteres.

La función `genvarname` es útil para crear nombres de variables que son válidas y únicas. Por ejemplo, `cadena=genvarname({'A', 'A', 'A', 'A'})` produce la siguiente salida: `cadena = 'A' 'A1' 'A2' 'A3`.

MATLAB puede trabajar con varios tipos de variables: en punto flotante, flotante doble, enteros, enteros dobles, enteros cortos o tipo char, cadenas de caracteres, expresiones simbólicas, etc. Los diversos tipos de variables son interpretados como matrices.

Para saber qué tipos de variables estamos usando podemos utilizar el comando `whos` como en los ejemplos siguientes, primero se declaran las variables y posteriormente se usa la función `whos`:

```
fx >>  i=9  ↩
       i=
```

9

```
fx >>  x=3.35  ↩
       x=
```

3.3500

```
fx >>  cadena= 'esta es una cadena de caracteres'  ↩
       cadena=
```

'esta es una cadena de caracteres'

```
fx >>  A=[2 3; 3 4]  ↩
```

```
A=
```

$$\begin{matrix} 2 & 3 \\ 3 & 4 \end{matrix}$$

```
fx >> whos ↩
=
```

Name	Size	Bytes	Class	Attributes
A	2×2	32	double	
cadena	1×32	64	char	
i	1×1	8	double	
x	1×1	8	double	

La columna `Size` indica la cantidad de memoria asignada a esa variable, mientras que la columna `Bytes Class` indica el tipo de variable. Obsérvese que **MATLAB** trata a todas las variables como matrices para propósitos de programación.

Al terminar una sesión, o cuando ya no se usen la variables debido a que se desea trabajar con otro programa es recomendable limpiar todas las variables para liberar memoria. Para realizar lo anterior, se puede hacer con cada variable, por ejemplo `clear i`, posteriormente `clear x` y finalmente `clear cadena`. Otra forma de realizarlo es limpiando simultáneamente las tres variables a través de: `clear i x cadena` o cuando hay una gran cantidad de variables, entonces es conveniente usar `clear all`.

1.4.2 Números

La notación convencional que usa **MATLAB** para la representación de números es la decimal con un punto decimal, signo $\pm$ y un número determinado de dígitos después del punto decimal, esto depende del tipo de formato numérico empleado (ver `format`). En la notación científica se usan potencias de diez. También se emplean constantes tales como e, π. Los números complejos o imaginarios emplean i o j. Todos los números son almacenados internamente utilizando el formato *long* en notación estándar de la IEEE para punto flotante con precisión finita de 16 dígitos decimales y un rango finito de 10^{-308} a 10^{308}.

Algunos ejemplos de números son:

$$\begin{array}{ccc} 2.33 & -105 & 0.005 \\ 9.999 & 8.345e88 & 9+i3.2 \\ 3.1416 & 8e5i & \frac{90}{180} \\ e^{\pi} & \text{sen}(\pi) & \sqrt{2} \end{array}$$

La tabla 1.2 muestra algunas de las constantes más usadas en el área de la ingeniería.

Tabla 1.2 Constantes útiles

pi	3.141592653589793.
Número imaginario `i`	$\sqrt{-1}$
`j`	Igual que el número imaginario `i`
eps	Punto flotante con precisión relativa, $\varepsilon = 2^{-52}$
`realmin`	El número en punto flotante más pequeño, 2^{-1022}
`realmax`	El número más grande en punto flotante, $(2-\varepsilon)^{1023}$
`inf`	Infinito ∞
`NaN`	No es un número

1.4.3 Formato numérico

La forma para desplegar funciones, variables y cualquier tipo de dato por **MATLAB** se realiza a través del comando `format` la cual controla el formato numérico de los valores desplegados. Es decir, modifica el número de dígitos para el desplegado de los datos. Este comando solo afecta a los números que son desplegados, no al proceso de cómputo numérico o al registro de las variables o datos.

```
fx >>   format short  ↩
fx >>  x= [ 9/8 8.3456e-8]  ↩
```

```
x=
```

$$1.1250 \quad 0.0000$$

Observe que en el caso de la constante 8.3456e-8 lo despliega con 0.0000 debido a los límites del formato corto `format short`.

Ahora note la diferencia con `format short e`

```
fx >>  format short e ↩
fx >>  x ↩
       x=
```

$$1.1250e+000 \quad 8.3450e-008$$

Es posible eliminar el formato extendido para desplegar el resultado de 9/8 y al mismo tiempo mantener la característica de desplegado para la segunda componente de la variable `x` de la siguiente forma:

```
fx >>  format short g ↩
fx >>  x ↩
       x=
```

$$1.1250 \quad 8.3450e-008$$

Para desplegar la información de `x` con mayor número de dígitos sin utilizar la notación científica, entonces

```
fx >>  format long ↩
       x=
```

$$1.125000000000000 \quad 0.000000083450000$$

Para saber qué tipo de `format` está actualmente activo se usa el comando `get`:

```
fx >>  get(0,'format') ↩
       ans=
```

long

El formato numérico que por default emplea **MATLAB** es el formato corto (*short*): `format short`. Sin embargo, es posible utilizar otros tipos de formatos como: extendido, notación científica, punto fijo, punto flotante, formato de ingeniería, etc. Los formatos numéricos que contiene el comando `format` se encuentran resumidos en la tabla 1.3.

Tabla 1.3 `Opciones del comando format`

Tipo	Descripción
`short`	Formato por default. Despliega 4 dígitos después del punto decimal. Por ejemplo 3.1416
`long`	Formato para punto fijo con 15 dígitos después del punto decimal para variables tipo doble y 7 dígitos para variables tipo simple. Por ejemplo x=4.123456789012345
`ShortE`	Formato para punto flotante con 4 dígitos después del punto decimal. Por ejemplo: 1.4567e+000. Variables enteras que almacenan números flotantes con más de 9 dígitos no serán desplegados en la notación científica.
`longE`	Punto flotante con 15 dígitos después del punto decimal para datos tipo doble y 7 dígitos después del punto decimal para variables tipo simple. Ejemplo x=4.123456789012345+e000. Variables enteras que almacenan números flotantes con más de 9 dígitos no serán desplegados en notación científica.
`shortG`	Formato para punto fijo o punto flotante, 4 dígitos después del punto decimal. Ejemplo 3.1416.
`longG`	Formato para punto flotante o fijo con 14 o 15 dígitos después del punto decimal para variables tipo doble y 6 o 7 dígitos después del punto decimal para variables tipo simple. Ejemplo 8.01234567891234.
`shortEng`	Formato para ingeniería tiene 4 dígitos después del punto decimal y una potencia que es múltiplo de tres. Ejemplo 3.1416e+000.
`longEng`	Formato para ingeniería tiene 16 dígitos después del punto decimal y una potencia que es múltipo de tres. Ejemplo 3.14159265358979e+000.

El comando `format` también proporciona representación hexadecimal, tipo proporción $\frac{a}{b}$ y de tipo bancaria como se explica en la tabla 1.4.

Tabla 1.4 `Opciones del comando format`

Tipo	Descripción
`rat`	Despliega en forma de proporción de números enteros. Ejemplo 17/50
`hex`	Representación en formato hexadecimal. Ejemplo a0fcbdf34
`bank`	Despliega tipo cantidad bancaria (pesos y centavos). Ejemplo 3.14

1.4.4 Operadores

Los operadores en **MATLAB** juegan un papel determinante ya que manipulan a las variables y funciones. Adicional a los operadores aritméticos de la tabla 1.1, hay operadores específicos para funciones, operaciones lógicas, relacionales, estructuras de datos, uniones y matrices.

A continuación se presentan diversos tipos de operadores de utilidad en **MATLAB**.

Operador colon :

El operador colon **:** (dos puntos verticales) es uno de los operadores más importantes para programar, se emplea para diferentes formas, como por ejemplo en **MATLAB** técnicamente se conoce como vectorización al proceso de generar una secuencia de número usando 1:10.

```
fx >>  1:10  ↩
       ans =

    1   2   3   4   5   6   7   8   9   10
```

En este caso el incremento es de uno en uno. Si se requiere un paso de incremento

específico, por ejemplo 2, entonces se procede de la siguiente forma

```
fx >>  1:2:10 ↩
       ans=
```

$$1 \quad 3 \quad 5 \quad 7 \quad 9 \quad 10$$

El valor del paso de incremento también puede ser menor que 1, por ejemplo

```
fx >>  1:0.1:2 ↩
       ans=
```

$$1 \quad 1.1 \quad 1.2 \quad 1.3 \quad 1.4 \quad 1.5 \quad 1.6 \quad 1.7 \quad 1.8 \quad 1.9 \quad 2$$

En ocasiones es conveniente que sea negativo

```
fx >>  10:-1:2 ↩
       ans=
```

$$10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2$$

En otras aplicaciones es útil realizar una variación desde -8 a 8 con incrementos de pasos de 0.5, por ejemplo

```
fx >>  -8:0.5:8 ↩
ans=
```

$$\begin{array}{cccccccccccccc} -8 & -7.5 & -7 & -6.5 & -6 & -5.5 & -5 & -4.5 & -4 & -3..5 & -3 & -2.5 & -2 \\ -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 & 2 & 2.5 & 3 & 3.5 & 4 & 4.5 \\ 5 & 5.5 & 6 & 6.5 & 7 & 7.5 & 8 \end{array}$$

Operador ()

Los operadores paréntesis **()** son utilizados en diversas operaciones matemáticas, por ejemplo: evaluar funciones como en el caso de `y=sin(t)`. También para referenciar a elementos de matrices $A(2,3)$. En expresiones aritméticas indican precedencia y la forma de agrupar variables `x=3*(y+j)*r-i*(4+r)`. Resultan de utilidad para evaluar condiciones lógicas `if (x>3) x=3 end`. Cuando se trata de evaluar potencias los paréntesis determinan los niveles de jerarquía para realizar operaciones desde las

más internas hasta las externas $(7+(r*(x^2+8))^6-j+\ \text{sen}(t^3)^5)^3$.

Operador semicolon ;

El operador punto y coma alineados en forma vertical o mejor conocido como semicolon **;** tiene varias funciones. Una de ellas se encuentra relacionada con el desplegar el resultado que tienen las variables, constantes, funciones o gráficas. Cuando se inserta al final de la expresión, instrucción o comando se inhabilita el desplegado. Por ejemplo:

```
fx >>  w=sin(pi/2);  ↩
fx >>
```

Si el operador **;** no se coloca al final de la instrucción, entonces se produce el desplegado del valor de la variable `w`

```
fx >>  w=sin(pi/2)  ↩
       w=
```

$$1$$

Otra funcionalidad del operador **;** es generar renglones en matrices. Por ejemplo,

```
fx >>  A=[ 19 3; 4 5 ]  ↩
       A=
```

$$\begin{matrix} 19 & 3 \\ 4 & 5 \end{matrix}$$

El operador **;** que precede al número 3 y antecede al número 4 genera un renglón de esta matriz. En este caso el primer renglón lo forman los números 19 y 3, y para el segundo renglón formado por 4 y 5. Obsérvese que después del corchete] no se inserta el operador **;** ya que en este caso afecta el desplegado, mientras que dentro de los corchetes [**;**] significa que genera un renglón en la matriz. Es decir el operador **;** tiene dos funciones muy diferentes. Por cada **;** dentro de corchetes se generará un renglón de la matriz. Es muy importante que quede claro la aplicación de este operador para evitar problemas de programación.

Si en el anterior ejemplo, en la matriz `A` no se insertara el operador **;** entonces el resultado quedaría de la siguiente forma:

```
fx >>  A=[ 19 3 4 5 ]  ↩
       A=
```

19 3 4 5

Para ver más detalles del operador **;** referirse a la sección de **Matrices y arreglos**.

Operador ,

El operador coma **,** tiene más de una función en MATLAB. Por ejemplo, cuando se emplea en funciones indica la separación de los argumentos como en el caso de `y=sin(t,x)` o en `w=sinh(t,x,y,z,p)`. Para referenciar a los elementos de una matriz se especifica el número de renglón y de la columna separados por una coma `A(3,4)`. En matrices indica la separación de los elementos de un renglón A=[5,6,7,8; 8,3,0,2].

Operador '

El operador ' se relaciona con el manejo de datos tipo char o cadena de caracteres. También representa la transpuesta de una matriz.

En relación a caracteres se emplea de la siguiente forma:

```
fx >>  dato_char= 'a'  ↩
       dato_char=
```

Para almacenar una cadena de caracteres, se procede de la siguiente forma:

```
fx >>  cadena= 'una cadena de letras'  ↩
       cadena=
```

una cadena de letras

El operador ' es ampliamente utilizado en funciones donde el pase de parámetros es a través de cadenas de texto como en `disp('Hola, mundo...')`.

En matrices el operador ' representa la matriz transpuesta, es decir A'=A^T. Para mayor detalle de este operador en matrices ver la sección de **Matrices y arreglos**.

Operador %

En el lenguaje de MATLAB se pueden insertar comentarios usando el operador %. Los comentarios son importantes en todo programa ya que permiten la documentación técnica del algoritmo o aplicación a implementar. Por ejemplo,

```
clc ; %limpia la pantalla de la ventana de comandos\
clear ; %limpia espacio de trabajo\
      ; %así como todas las variables que ocupen memoria
t=0:0.1:t; %genera base de tiempo
y=sin(t); %genera el vector senoidal
plot(t,y) %gráfica  onda senoidal
```

Operador ~

El operador *tilde* ~ se emplea para deshabilitar una variable de salida de una función. Es muy útil cuando la función retorna más de una variable y no se requiere usar todas las variables; supóngase que la función `control_robot` retorna dos variables (error y par) y la sintaxis es: `[error, par ]=control_robot(q)`. No se requiere usar la variable `error`, únicamente `par`, entonces se usa de la forma siguiente:

`[~ , par]= control_robot(q)`.

También se emplea como negación en operadores lógicos, por ejemplo en ~ = que significa `no es igual a`.

En las siguientes secciones se irán explicando otros tipos de operadores que se encuentran directamente relacionados con matrices y en general con la programación del lenguaje de MATLAB.

1.5 Matrices y arreglos

En **MATLAB** una matriz es un arreglo rectangular de datos o números, tienen n renglones por p columnas; la notación matemática más común para representar a una matriz es: $A \in \mathbb{R}^{n\times p}$. Matrices con una sola columna o renglón significan vectores para **MATLAB**, por ejemplo $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ o $\boldsymbol{y} \in \mathbb{R}^{1\times n}$, respectivamente. Especial significado representan los escalares cuya interpretación para propósitos de programación corresponde a una representación de matriz del tipo $\mathbb{R}^{1\times 1}$.

Las entradas de la matriz se conocen como *elementos* y pueden ser números reales, números complejos, funciones, operadores, inclusive también pueden ser matrices de menor dimensión. Es importante destacar que una matriz es un objeto matemático por sí misma, esto es, no representa un número o un escalar. Sin embargo, en lenguaje **MATLAB** es posible trabajar a una matriz como un escalar, por ejemplo $\alpha \in \mathbb{R}^{1\times 1}$. La matrices tienen operaciones y propiedades bien definidas. Las operaciones entre matrices producen una matriz, en contraste operaciones entre vectores pueden producir un escalar, vector o matriz.

En la figura 1.4 se muestra la representación clásica de una matriz rectangular de n renglones por p columnas (matriz n por p). Las columnas son arreglos verticales. Por ejemplo, la segunda columna está formada por los elementos $A(1,2), A(2,2), \cdots, A(n,2)$ mientras que los renglones son filas horizontales dentro del arreglo rectangular; el segundo renglón está compuesto por $A(2,1), A(2,2), \cdots, A(2,p)$. Las columnas se incrementan de izquierda a derecha, mientras que los renglones se incrementan de arriba hacia abajo.

$$A = \begin{bmatrix} A(1,1) & A(1,2) & \cdots & A(1,p) \\ A(2,1) & A(2,2) & \cdots & A(2,p) \\ \vdots & & \ddots & \vdots \\ A(n,1) & A(n,2) & \cdots & A(n,p) \end{bmatrix}$$

Columna

Renglón

Figura 1.4 Componentes de una matriz: elementos $A(i,j)$, renglones y columnas.

Los elementos de una matriz se denotan en **MATLAB** por $A(i,j)$ donde $i = 1 \ldots n$ representa el i-ésimo renglón y $j = 1 \ldots p$ denota la j-ésima columna. Por ejemplo, en la matriz $A \in \mathbb{R}^{5\times 5}$ el elemento $A(2,5)$ significa la componente del segundo renglón y quinta columna, el cual se muestra en la siguiente expresión:

$$A = \begin{bmatrix} A(1,1) & A(1,2) & A(1,3) & A(1,4) & A(1,5) \\ A(2,1) & A(2,2) & A(2,3) & A(2,4) & \boxed{\text{A(2,5)}} \\ A(3,1) & A(3,2) & A(3,3) & A(3,4) & A(3,5) \\ A(4,1) & A(4,2) & A(4,3) & A(4,4) & A(4,5) \\ A(5,1) & A(5,2) & A(5,3) & A(5,5) & A(5,5) \end{bmatrix}.$$

En esta matriz en particular los pivotes pueden adquirir valores para referenciar a los elementos de la matriz $A(i,j)$. Para los renglones $i = 1, 2, \cdots, 5$ y en las columnas $j = 1, 2 \cdots, 5$.

Para los propósitos de identificar los elementos de una matriz, éstos pueden ser vistos sobre un sistema de referencia cartesiano planar cuyo origen se encuentra localizado en la esquina superior izquierda. Las referencias a elementos se hace con $A(i,j)$. Es importante aclarar que dicho origen no inicia en $A(0,0)$; lo que sería un error de sintaxis. El origen inicia con los pivotes asignados en uno, como en $A(1,1)$; con esta asignación se hace referencia al primer elemento de la matriz, ubicado en el primer renglón y primera columna. El elemento $A(1,1)$ representa el origen o punto de partida para los pivotes (i,j) donde el índice del i-ésimo renglón está dado antes del j-ésimo índice de la columna. Los renglones de una matriz son numerados de arriba hacia abajo y las columnas de izquierda a derecha.

Se debe tener claro que el lenguaje **MATLAB** no admite referencias a elementos de una matriz con pivotes (i,j) asignados en cero, negativos, números reales o números enteros de magnitud mayor a la dimensión de la matriz. En otras palabras, en las referencias a elementos de la matriz únicamente con números enteros o naturales dentro de los límites de la dimensión de la matriz. Por ejemplo, los siguientes elementos $A(0,0), A(0,5), A(-5,-1), A(5,0)$ son referencias inválidas de una matriz.