ITAM

Una Descripción de MatLab

Mauricio García Esteban *

Versión 1
enero de 1998

* *ITAM Instituto Tecnológico Autónomo de México*, Centro de Cómputo

Contenido

Slide 1

Definición de MATLAB versión 5.1

- MATLAB, es un laboratorio interactivo de Matrices.
- Aplicación a Cálculos científicos e Ingeniería.
- Resuelve problemas númericos complejos sin escribir un programa en realidad.
- MATLAB, es una abreviatura para MATrix LABoratory.

1 Introducción.

MATLAB es un sistema interactivo basado en matrices para cálculos científicos y de ingeniería. Se pueden resolver problemas numéricos relativamente complejos sin escribir un programa en realidad. El nombre de MATLAB es una abreviatura para MATrix LABoratory.

1.1 Objetivo

El propósito de estas notas es ayudar en la iniciación a MATLAB. La mejor forma de utilizarlas es poner manos a la obra. Se aconseja leer las notas, realizar sus anotaciones y ha trabajar con la computadora realizando los ejercicios planteados.

Slide 2

Iniciando MATLAB

- Instrucción **Help**
- Instrucción **Help general**
- Instrucción **Help path**
- Instrucción **lookfor**
- Instrucción **demo**

Después de entrar en MATLAB en la forma que se explica en la sección ??, se puede utilizar la ayuda de la instrucción `help`, para solicitar información. la instrucción `help` mostrará una lista de modulos, para las que se puede obtener ayuda mientras se está trabajando; la instrucción `help` *nombre_del_modulo* nos dará información sobre las funciones específicas, de ese modulo. Así, la instrucción `help general`, nos dará información sobre las funciones del modulo `general`. Especificamente podemos solicitar información, de una función, por ejemplo `help eig` que calcula los autovalores de una matriz. Se pueden ver algunas de las capacidades de MATLAB usando la instrucción `demo`.

Actualmente el ITAM tiene MatLab 5.1, para el trabajo mediante la red, con los siguientes modulos:

Neuronal Network, Control Systems, System Identification, Signal Procesing, Nag Foundation, Wavelet, Simulink, Communications, DSP Blockset, Image Processing, y una licencia para Statistics, Optimization, Financial.

El alcance y la potencia de MATLAB van más allá de lo que podemos ver en estas notas. En algún momento puede desear una información más detallada. Es el momento de consultar la guía del usuario y la de referencia.

Slide 3

```
matlab“general      - General purpose commands.
matlab“ops          - Operators and special characters.
matlab“lang         - Programming language constructs.
matlab“elmat        - Elementary matrices and matrix manipulation.
matlab“elfun        - Elementary math functions.
matlab“specfun      - Specialized math functions.
matlab“matfun       - Matrix functions - numerical linear algebra.
matlab“datafun      - Data analysis and Fourier transforms.
matlab“polyfun      - Interpolation and polynomials.
matlab“funfun       - Function functions and ODE solvers.
matlab“sparfun      - Sparse matrices.
matlab“graph2d      - Two dimensional graphs.
matlab“graph3d      - Three dimensional graphs.
matlab“specgraph    - Specialized graphs.
matlab“graphics     - Handle Graphics.
matlab“uitools      - Graphical user interface tools.
matlab“strfun       - Character strings.
matlab“iofun        - File input/output.
matlab“timefun      - Time and dates.
matlab“datatypes    - Data types and structures.
matlab“dde          - Dynamic data exchange (DDE).
matlab“demos        - Examples and demonstrations.
comm“comm           - Communications Toolbox
comm“commsfun       - Communications Toolbox SIMULINK S-functions.
comm“commsim        - Communications Toolbox SIMULINK files.
wavelet“wavelet     - Wavelet Toolbox.
wavelet“wavedemo    - Wavelet Toolbox Demos.
nag“nag             - NAG Foundation Toolbox - Numerical & Statistical Library
nag“examples        - NAG Foundation Toolbox - Numerical & Statistical Library
dspblks“dspblks     - DSP Blockset.
dspblks“dspmex      - (No table of contents file)
dspblks“dspdemos    - (No table of contents file)
images“images       - Image Processing Toolbox.
images“imdemos      - Image Processing Toolbox — demos and sample images
nnet“nnet           - Neural Network Toolbox.
nnet“nndemos        - Neural Network Demonstrations and Applications.
toolbox“signal      - Signal Processing Toolbox.
toolbox“ident       - System Identification Toolbox.
toolbox“control     - Control System Toolbox.
control“obsolete    - (No table of contents file)
simulink“simulink   - Simulink
simulink“blocks     - Simulink block library.
simulink“simdemos   - Simulink demonstrations and samples.
simulink“dee        - Differential Equation Editor
toolbox“local       - Preferences.
```

Slide 4

Temario

- Acceso a MATLAB
- Introducción de matrices
- Operaciones con matrices, operaciones a coordenadas
- Declaraciones, expresiones, variables; almacenamiento de sesiones
- Funciones para la construcción de matrices
- For, while, if — y relaciones
- Funciones escalares
- Funciones vectoriales
- Funciones matriciales
- Comandos de edición de línea y rellamada
- Submatrices y notación de dos puntos
- Archivos .m
- Cadenas de texto, mensajes de error, input
- Tratamiento de archivos .m
- Comparación de la eficiencia de algoritmos: flops y etime
- Formato de salida
- Hardcopy
- Gráficos
- Consulta

Slide 5

Iniciando MATLAB

- c:\matlab\bin\matlab.exe
- >>quit

1.2 iniciando MATLAB.

Después de entrar a la mayoría de los sistemas, para acceder a MATLAB basta utilizar la instrucción `matlab` y para salir, la instrucción `exit` o `quit`. Por ejemplo si estamos en un PC, salvo que tengamos el programa en un directorio aparte, basta con escribir

```
C:\matlab\bin>  matlab.exe
```

Podemos salir de él con la instrucción:

```
>>  quit
```

Slide 6

$$\boxed{\textit{variable} = \textit{expresión}, \textbf{ o simplemente } \textit{valor}}$$

- Las expresiones se componen, normalmente, a partir de operadores, funciones y nombres de variables. La evaluación de una expresión produce una matriz, que se muestra en pantalla, y se asigna a la variable para su posterior uso. Si se omiten la variable y el signo =, se crea una variable llamada `ans` (por answer) a la que se asigna el resultado de la expresión.
- instrucción `who`
- instrucción `clear`

1.3 Declaraciones.

MATLAB distingue las letras mayúsculas de las minúsculas en los nombres de instrucciones, funciones y variables. Así, `resolvente` no es lo mismo que `ReSoLvEnTe`.

La instrucción `who` muestra las variables que se encuentran en el espacio de trabajo. Para eliminar una variable de la memoria se utiliza la instrucción `clear` *nombre_variable*. Si se escribe sólo `clear` se borran todas las variables no permanentes.

La variable permanente `eps` (épsilon) da la precisión de la máquina—alrededor de 10^{-16} en la mayoría de ellas. Es útil para determinar la tolerancia en procesos iterativos.

Cualquier tipo de cálculo, gráfico, o impresión puede detenerse sin salir del programa con CTRL-C (CTRL-BREAK en PC).

Slide 7

Matrices en MATLAB

- `help matfun`
- Formas de Introducir una matriz en MATLAB
  - Introduciendo una lista explícita de elementos,
  - Generándola mediante funciones y declaraciones, ver la sección 3.5
  - Creándola en un archivo .m (ver la sección 7.1),
  - Cargándola de un archivo de datos externo

2 Vectores y matrices

MATLAB trabaja esencialmente con un solo tipo de objetos: una matriz numérica rectangular(ó arreglos multidimensionales), con entradas posiblemente complejas; todas las variables representan matrices. A veces, las matrices 1×1 se consideran escalares, y las matrices con una sola fila o columna se consideran como vectores.

Hay varias formas diferentes para introducir una matriz en MATLAB. A saber:

- Introduciendo una lista explícita de elementos,
- Generándola mediante funciones y declaraciones,
- Creándola en un archivo .m (ver sección 7.1),
- Cargándola de un archivo de datos externo

Slide 8

Matrices en MATLAB

Por ejemplo, cualquiera de las declaraciones

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

y

```
>> A = [
          1  2  3
          4  5  6
          7  8  9 ]
```

crea la matriz 3×3 que se espera y la asigna a una variable A. Inténtelo. Los elementos en una fila de una matriz pueden separarse tanto por comas como por espacios en blanco.

2.1 Declaración.

Cuando alguno de los números se escribe en forma exponencial (por ejemplo 2.34e-9), deben evitarse los espacios en blanco. La escritura de una matriz grande debe hacerse preferentemente en un archivo .m, donde es más sencillo corregir errores (ver sección 7.1).

Slide 9

Matrices en MATLAB

```
>> x=[4/3 1.234e-6]

x =
    1.3333    0.0000
>> format long e
>> x

x =
    1.333333333333333e+000    1.234000000000000e-006
```

2.2 Formato de visualización.

El commando `format` controla el formato numérico de los valores desplegados sobre el monitor. El comando, tan sólo afecta el desplegado en el monitor, y no en los cálculos efectuados por MATLAB.

- short
- short e
- short g
- long
- long e
- long g
- bank
- rat
- hex

Slide 10

Matrices en MATLAB

```
>> A=magic(100);

>> s=1-1/2+1/3-1/4+1/5-1/6+1/7 ...
     -1/8+1/9-1/10+1/11-1/12

s =

    6.532106782106782e-001
```

2.3 Suprimir la visualización a la salida.

Si se teclea la declaración y presionar la tecla "enter", MATLAB automáticamente despliega los resultados sobre la pantalla. Sin embargo, si se termina la linea con punto y coma, no desplegará, los cálcaulos intermedios ni el resultado final. Esto es particularmente útil cuando se generan enormes cálculos y se desea optimar el tiempo de proceso.

2.4 Largas lineas de comando

Si la declaración es demasiado larga y no es suficiente la línea de comandos, utilice tres puntos consecutivos, seguido por la tecla "enter", para indicar que se continuará con la declaración en la siguiente línea.

Slide 11

Hoja de trabajo en MATLAB

- `save june10`
- `save june10 x y z`
- `load june10`
- `load filename -mat`
- `save filename x*`
- `load filename ex1*95`

2.5 Salvar y cargar la hoja de trabajo.

MATLAB tiene los comandos `save` y `load`, quienes permiten salvar los contenidos de la hoja de trabajo de MATLAB, en cualquier momento durante la sesión, y retomar los datos en MATLAB, durante la misma sesión o alguna otra más tarde.

2.6 Especificando el formato.

Se puede controlar el salvar los datos en diferentes tipos de formato:

- -mat, formato en forma binaria, default.
- -ascii, formato en código ASCII de 8 digitos.
- -ascii -double, código ASCII de 16 digitos.
- -ascii -double -tabs, delimitar los elementos del arreglo mediante tabs
- -v4, salvar en la versión 4 de MATLAB
- -append, datos para archivos Mat

Funciones de Algebra Lineal. directorio `matfun`.

categoria	Función	Descripción
Análisis	norm	Norma de la matriz
	normest	Estima la norma-2 de la matriz
	rank	rango de la matriz
	det	determinante de la matriz
	trace	suma de los elementos diagonales
	null	espacio nulo
	orth	Ortogonalización.
	rref	reduce a la forma de renglon echelon
	subspace	ángulo etre dos subespacios
Ecuaciones Lineales	\ y /	solución de la ecuación lineal
	inv	inversa de la matriz
	con	número de condición para inversión
	condest	1-norma estimado
	chol	factorización Cholesky
	cholinc	factorización incompleta Cholesky
	lu	factorización LU
	luinc	factorización LU, incompleta
	qr	descomposición Ortogonal-triangular
	nnls	minimos cuadrados nonegativo
	pinv	seudoinversa
	lscov	minimos cuadrados con covariancia

Slide 12

Slide 13

Funciones de Algebra Lineal, Continuación. directorio `matfun`.

categoria	**Función**	**Descripción**
Valores Propios y valores propios	eig	valores propios y valores propios
	svd	descomposición por valor singular
	eigs	algunos valores propios
	svds	algunos valores singulares
	poly	polinomio característico
	polyeig	problema de v.p en polinomios
	condeig	número de condición para v.p.
	hess	forma Hessenberg
	qz	factorización QZ
	schur	descomposición de Schur
Funciones de Matriz	expm	matriz exponencial
	logm	matriz logaritmica
	sqrtm	raíz cuadrada de la matriz
	funm	evaluación general en matriz

Slide 14

```
>> A=pascal(3)
A =
     1     1     1
     1     2     3
     1     3     6
>> B=magic(3)
B =
     8     1     6
     3     5     7
     4     9     2
>> C=fix(10*rand(3,2))
C =
     9     4
     2     8
     6     7
```

```
>>A(2,3)=2
A =
     1     1     1
     1     2     2
     1     3     6

>> A(2,3)=3;
 u=[3;1;4]
u =
     3
     1
     4
>> v=[2 0 -1]
v =
     2     0    -1
>> s=7;
```

Las funciones internas `rand`, `magic`, y `hilb`, por ejemplo, proporcionan una forma sencilla para crear matrices con las que experimentar. La instrucción `rand(n)`, resp. `rand(m,n)`, creará una matriz $n \times n$, resp. $m \times n$, con entradas aleatoriamente generadas, distribuidas uniformemente entre 0 y 1. `magic(n)` creará una matriz cuadrada mágica (las filas y las columnas suman la misma cantidad) con entradas enteras; `hilb(n)` creará la matriz de Hilbert de orden n, la reina de las matrices mal condicionadas. m y n, por supuesto, denotan enteros positivos. También se pueden crear matrices utilizando bucles `for`. Inténtelo.

Las entradas individuales de una matriz o de un vector se pueden obtener poniendo los índices entre paréntesis de la forma usual. Por ejemplo, $A(2,3)$ denota la entrada en la segunda fila y tercera columna de la matriz A y $x(3)$ denota la tercera coordenada del vector x. Inténtelo. Sólo se pueden usar como índices de vectores y de matrices enteros *positivos*.

Slide 15

Operaciones

$+$	adición
$-$	sustracción
$*$	multiplicación
^	potenciación
$'$	traspuesta
$\backslash$	división izquierda
$/$	división derecha

```
>> X=A+B
X =
     9     2     7
     4     7    10
     5    12     8
>> Y=X-A
Y =
     8     1     6
     3     5     7
     4     9     2
>> x=v*u;
```

3 Operaciones con Matrices

Estas operaciones para matrices se aplican también a escalares (matrices 1×1). Si los tamaños de las matrices son incompatibles para la operación matricial se obtiene un mensaje de error, exceptuando el caso en que uno de los operandos sea un escalar y el otro una matriz (para la adición, sustracción, división y multiplicación). En esta situación se opera el escalar con cada término de la matriz.

Slide 16

división.

$x = A\backslash b$ es la solución de $A * x = b$ y, resp.,

$x = b/A$ es la solución de $x * A = b$.

La "división matricial" merece un comentario especial. Si A es una matriz invertible y b es una columna, resp. fila, compatible, entonces

$x = A\backslash b$ es la solución de $A * x = b$ y, resp.,

$x = b/A$ es la solución de $x * A = b$.

En la división izquierda, si A es cuadrada, se factoriza utilizando eliminación gaussiana. Con los factores se resuelve $A * x = b$. Si la matriz A no es cuadrada, se factoriza utilizando la ortogonalización de Householder con pivoteo de columnas. Con los factores se resuelve el sistema indeterminado o sobredeterminado en el sentido de los mínimos cuadrados. La división derecha se define a partir de la izquierda por $b/A = (A'\backslash b')'$.

Slide 17

```
>> x=v*u
x =
     2
>> X=u*v
X =
     6     0    -3
     2     0    -1
     8     0    -4
>> X=B'
X =
     8     3     4
     1     5     9
     6     7     2
```

```
>> z=[1+2i 3+4i]
z =
   1 + 2i   3 + 4i
>> z'
ans =
   1 - 2i
   3 - 4i
>> z.'
ans =
   1 + 2i
   3 + 4i
```

3.1 Producto de vectores y transpuesta

Un vector renglón y un vector columna de la misma longitud pueden ser multiplicados en cualquier orden.

Para matrices reales, la operación *transpuesta* intercambia a_{ij} y a_{ji}, MATLAB utiliza el apóstrofe (o acento simple) para denotar la transpuesta. Realice $X = B'$, y $X = A'$, se dará cuenta que la transpuesta de A, es la matriz A, esto se debe a que se trata de una matriz simétrica, y sin embargo $B \neq B'$

Para un vector o matriz con complejos, la cantidad z' denota la *transpuesta conjugada compleja*. La transpuesta no conjugada compleja es denotada por $z.'$, en analogía con las operaciones con arreglos.

Slide 18

```
>> X=A*B
X =
    15    15    15
    26    38    26
    41    70    39
>> Y=B*A
Y =
    15    28    47
    15    34    60
    15    28    43
```

```
>> for i=1:m
for j=1:n
C(i,j)=A(i,:)*B(:,j);
end
   end
```

3.2 Multiplicación de matrices

MATLAB, denota con el asterisco la operación de multiplicación. La matriz producto $C = BA$, es definida cuando la dimensión de la columna de A es igual a la dimensión del renglón de B, o cuando uno de ellos es escalar. Si A es una mariz m *por* n y B es una matriz p *por* n, su producto C, es m *por* n. El producto puede ser definido en MATLAB usando loops, notación de dos puntos, y producto punto de vectores.

Slide 19

```
>> A=[0 -6 -1 ; ...
  6 2 -16; -5 20 -10]
A =
     0    -6    -1
     6     2   -16
    -5    20   -10
>> X=A^2
X =
   -31   -32   106
    92  -352   122
   170  -130  -215
>> A=sqrtm(X);
```

```
>> for t=0:0.01:1
   X=[X expm(t*A)*x0];
   end
>> plot3(X(1,:),X(2,:),...
X(3,:),'-o')
```

3.3 Potencias y exponenciales con Matrices.

La matriz A debe ser cuadrada para realizar la potenciación. Si A es no singular el exponente puede ser negativo. Se permite que el exponente sea una fracción cumpliendo la condición de existencia de la inversa de A, `inv(A)`.

El ejemplo paralelo define el cálculo de la solución $x(t)$, para la ecuación diferencial $x(t) = \exp^{tA} x(0)$ en 101 puntos sobre el intervalo $0 \leq t \leq 1$, y se visualiza en el plano de fase tridimensional. El objetivo es utilizar la instrucción *matriz exponencial* `expm()`

Norma

Slide 20

```
>> [norm(v,1) norm(v,2)...
norm(v,inf)]
ans =
    3.0  2.2361  2.0
```

```
>> [norm(C,1) norm(C,2)...
norm(C,inf)]
ans =
    19.0  14.8015  13.00
```

3.4 Norma de vectores y Matrices

La norma del vector x, se define como:

$$||x||_p = \left(\sum x_i^p\right)^{\frac{1}{p}}$$

y en MATLAB se calcula mediante la instrucción `norm(x,p)`, donde $p > 1$ necesariamente, sin embargo lo más común es $p = 1, 2$ y ∞. El valor por default es $p = 2$, lo cual corresponde a la *distancia euclidiana*

La norma-p de una matriz A es:

$$||A||_p = \frac{\max}{x} ||A\,x||_p / ||x||_p$$

Slide 21

Construcción de Matrices.

Las siguientes funciones están disponibles en MATLAB:

Función.	**Descripción.**
eye	matriz identidad
zeros	matriz de ceros
ones	matriz de unos
diag	ver más adelante
triu	parte triangular superior de una matriz
tril	parte triangular inferior de una matriz
rand	matriz generada aleatoriamente
hilb	matriz de Hilbert
magic	matriz mágica
toeplitz	ver `help toeplitz`

3.5 Funciones para la construcción de matrices.

Por ejemplo, `zeros(m,n)` produce una matriz nula $m \times n$, y `zeros(n)` produce otra cuadrada de orden n; si A es una matriz, entonces `zeros(A)` produce una matriz de ceros del mismo orden que A.

Si x es un vector, `diag(x)` es la matriz diagonal con x en su diagonal; si A es una matriz cuadrada, `diag(A)` es un vector formado por la diagonal de A. ¿Qué será entonces `diag(diag(A))`? Inténtelo.

Las matrices se pueden construir por bloques. Por ejemplo, si A es 3×3, entonces

```
B = [A, zeros(3,2); zeros(2,3), eye(2)]
```

dará una cierta matriz 5×5. Inténtelo.

Slide 22

operaciones a coordenadas.

Es importante observar que las operaciones, $*$, $\hat{}$, \, y /, pueden evaluarse por coordenadas, y en MATLAB es factible al ir precedidas de un punto.

Por ejemplo, tanto $[1, 2, 3, 4]. * [1, 2, 3, 4]$ como $[1, 2, 3, 4].\hat{}2$ darán $[1, 4, 9, 16]$.

3.6 Operaciones a coordenadas.

Las operaciones de adición y sustracción operan intrínsecamente a coordenadas pero no todas las operaciones matriciales: Son operaciones *matriciales a coordenadas.* MATLAB cuenta con esta posibilidad, y son particularmente útiles cuando se utilizan las herramientas gráficas.

Slide 23

for

Las instrucciones:

```
x = []; for i = 1:n, x=[x,i^2], end
```

ó

```
x = [];
for i = 1:n
   x = [x,i^2]
end
```

darán como resultado un vector, mientras que

```
x = []; for i = n:-1:1, x=[x,i^2], end
```

dará el mismo vector en orden inverso.

4 Sentencias de control For, while, if — y relaciones

Básicamente, las instrucciones para el control de flujo de MATLAB operan como en la mayor parte de los lenguajes usuales.

4.1 for

Las instrucciones siguientes, en base al for, producirán e imprimirán en pantalla la matriz de Hilbert $m \times n$.

```
   for i = 1:m
   for j = 1:n
      H(i, j) = 1/(i+j-1);
   end
end
H
```

El punto y coma de la instrucción interior suprime la impresión no deseada de los resultados intermedios mientras que el último `H` muestra el resultado final.

while.

Slide 24

La forma general de un bucle `while` es:

`while` *relación*

  instrucciones

`end`

Las instrucciones se repetirán mientras la relación sea cierta.

4.2 while

Por ejemplo, dado un número a, las instrucciones siguientes calculan y muestran el menor entero no negativo n tal que $2^n \geq a$:

```
   n = 0;
while 2^n < a
   n = n + 1;
end
n
```

Slide 25

if

La forma general de un bucle `if` simple es:

`if` *relación*
    instrucciones
`end`

Las instrucciones se ejecutarán sólo si la relación es cierta.

4.3 If

También son posibles las ramificaciones múltiples, como se ilustra con

```
   if n < 0
   paridad = 0;
elseif rem(n,2) == 0
   paridad = 2;
else
   paridad = 1;
end
```

Si sólo tenemos dos ramificaciones podemos omitir, desde luego, la porción correspondiente a elseif.

Slide 26

relaciones

Los operadores relacionales en MATLAB son:

$<$	menor que
$>$	mayor que
$<=$	menor o igual que
$>=$	mayor o igual que
$==$	igual
$\sim=$	no igual.

Las relaciones pueden conectarse o cuantificarse por los operadores lógicos

&	y
$\vert$	o
$\sim$	no.

4.4 Relaciones

En el uso de las relaciones y operadores en MATLAB, recordemos lo siguiente:

- se usa "=" en las asignaciones mientras que para las relaciones se usa "==".
- Cuando se aplican a escalares los operadores lógicos, una relación es realmente el escalar 1 ó 0 dependiendo de si la relación es verdadera o falsa:

  Pruebe con `3 < 5`, `3 > 5`, `3 == 5` y `3 == 3`.
- Cuando se aplica a matrices del mismo orden, una relación entre ellas da lugar a una matriz de ceros y unos, dando el valor de la relación entre las correspondientes entradas.

  Pruebe con `a = rand(5), b = triu(a), a == b`.

Slide 27

Relación entre matrices.

Si se quiere ejecutar *algo*, cuando las matrices A y B sean iguales, se puede escribir:

```
if A == B
    algo
end
```

Si se desea ejecutar la instrucción *algo* cuando A y B son distintas:

```
if any(any(A ~= B))
    algo
end
```

o, más simplemente,

```
if A == B else
    algo
end
```

4.5 While e if con matrices.

Cuando se utiliza una relación entre matrices en un bucle `while` o `if`, la relación se entiende verdadera, si cada una de las entradas de la matriz de relación es no nula.

En el ejemplo denotemos que no se puede recurrir a la obviedad,

`if A ~= B,` *algo*`, end`

ya que no hará lo que deseamos. La instrucción sólo se ejecutará si *todas* las entradas de A son distintas de las de B y se requieren dos `any`s ya que `any` es un operador vectorial.

Las funciones `any` y `all` pueden utilizarse de forma creativa para reducir relaciones entre matrices a relaciones entre vectores y escalares.

La instrucción `for` permite usar cualquier matriz en vez de `1:n`. Ver la Guía del usuario para los detalles de cómo esta posibilidad amplía la potencia de la instrucción `for`.

funciones escalares

Slide 28

Las funciones más comunes entre estas son:

sin	*asin*	*exp*	*abs*	*round*
cos	*acos*	*log(natural)*	*sqrt*	*floor*
tan	*atan*	*rem(resto)*	*sign*	*ceil*

5 Funciones

5.1 Funciones escalares

Algunas funciones de MATLAB operan esencialmente sobre escalares, aunque lo hacen también sobre matrices (elemento a elemento).

Slide 29

Funciones vectoriales

max	*sum*	*median*	*any*
min	*prod*	*mean*	*all*
sort		*std*	

5.2 Funciones vectoriales

Otras funciones de MATLAB operan fundamentalmente sobre vectores (fila o columna), aunque también pueden operar sobre matrices $m \times n$ ($m \geq 2$) haciendolo en este caso columna a columna, produciendo, por tanto, un vector fila que contiene el resultado de su aplicación a cada columna. Para conseguir que actúen por filas basta usar la traspuesta; por ejemplo, `mean(A')'`. Veamos algunas de estas funciones:

Por ejemplo, la entrada máxima de un matriz A se obtiene con `max(max(A))` en vez de `max(A)`. Inténtelo.

Slide 30

eig	autovalores y autovectores
chol	factorización de Cholesky
svd	descomposición en valores singulares
inv	inversa
lu	factorización LU
qr	factorización QR
hess	forma de Hessenberg
schur	descomposición de Schur
rref	forma escalonada reducida por filas
expm	matriz exponencial
sqrtm	matriz raíz cuadrada
poly	polinomio característico
det	determinante
size	tamaño
norm	norma 1, norma 2, norma de Frobenius, norma ∞
cond	número de condición en la norma 2
rank	rango

5.3 Funciones matriciales

Las funciones de MATLAB admiten argumentos de salida simples o múltiples. Por ejemplo, `y = eig(A)`, o simplemente `eig(A)` genera un vector columna conteniendo los autovalores de A mientras que `[U,D] = eig(A)` produce una matriz U cuyas columnas son los autovectores de A y una matriz diagonal D con los autovalores de A en su diagonal. Pruebe.

Slide 31

submatrices

La expresión `1:5` (que ya encontramos en los bucles `for`) es realmente un vector fila: el `[1 2 3 4 5]`.

Los números no tienen que ser enteros ni el incremento uno. Por ejemplo, `0.2:0.2:1.2` da como resultado `[0.2 0.4 0.6 0.8 1.0 1.2]`, mientras que con `5:-1:1` se obtiene el vector `[5 4 3 2 1]`.

Las siguientes instrucciones, por ejemplo, generarán una tabla de senoides.

```
x = [0.0:0.1:2.0]';
y = sin(x);
[x y]
```

Hagamos notar que al operar `sin` a coordenadas, produce un vector y a partir de x.

6 Submatrices y notación de dos puntos

Los vectores y submatrices son utilizados a menudo en MATLAB para conseguir efectos de manipulación bastante complejos.

La "notación de dos puntos" (que se utiliza para generar vectores y submatrices), y la indexación por vectores, son las llaves para una manipulación eficiente de estos objetos. Su uso de forma creativa permite minimizar el número de bucles (que ocasiona una ejecución lenta del programa hecho en MATLAB) y hacen que las instrucciones sean más simples y legibles. *Debe hacerse un esfuerzo especial para familiarizarse con esta notación.*

Slide 32

•Dos puntos sin más especificación denotan una fila o columna completa:

`A(:,3)` es la tercera columna de A, y `A(1:4,:)` son las cuatro primeras filas.

•Se pueden usar como índices de vectores enteros arbitrarios:

`A(:,[2 4])` está formada por las columnas segunda y cuarta de A.

•Se pueden usar a ambos lados de una instrucción de asignación:

`A(:,[2 4 5]) = B(:,1:3)` reemplaza las columnas 2, 4 y 5 de A por las tres primeras de B. Realice la muestra y asigne la matriz A alterada *completa*.

•Las columnas 2 y 4 de A pueden multiplicarse por la derecha por una matriz 2×2:

`A(:,[2,4]) = A(:,[2,4])*[1 2;3 4]`

De nuevo muestre y asigne la matriz completa.

La notación de dos puntos permite acceder a submatrices. Por ejemplo, `A(1:4,3)` es el vector columna con las cuatro primeras entradas de la tercera columna de A.

Si denotamos por x un vector con n componentes, ¿cuál es el efecto de la instrucción `x = x(n:-1:1)`? Haga la prueba.

Para comprobar la utilidad de esta notación, podría comparar estas instrucciones de MATLAB con una rutina de Pascal, FORTRAN, o C que dé los mismos resultados.

Slide 33

archivo de instrucciones

Los archivos de instrucciones son utilizados a menudo para introducir datos en una matriz grande. En un archivo de este tipo es bastante sencillo corregir los errores sin tener que repetir todo el trabajo. Si, por ejemplo, se escribe en el archivo `datos.m` la secuencia:

```
A = [
1 2 3 4
5 6 7 8
];
```

entonces la instrucción de MATLAB `datos`, hará que se efectúe la asignación especificada en `datos.m`.

Un archivo .m puede hacer referencia a otros, incluyendo a él mismo.

7 Archivos .m

MATLAB puede ejecutar una sucesión de instrucciones almacenadas en archivos. Estos archivos se denominan "archivos .m", debido a que su sufijo debe ser "m". Gran parte del trabajo con MATLAB será el de crear y refinar archivos .m.

Hay dos tipos de archivos .m: *archivos de instrucciones* y *archivos de funciones.*

7.1 Archivos de instrucciones.

Un archivo de instrucciones consiste en una sucesión de instrucciones normales de MATLAB. Si tuviéramos un archivo denominado `nombre.m`, las instrucciones del archivo pueden ser ejecutadas sin más que escribir la instrucción `nombre`. *Las variables en un archivo de instrucciones son globales y, por tanto, cambiarán los valores del espacio de trabajo.*

Slide 34

archivo de funciones

```
function a = ental(m,n)
%ENTAL Matriz entera generada aleatoriamente.
%    ental(m,n) produce una matriz mxn con entradas
%    enteras entre 0 y 9
a = floor(10*rand(m,n));
```

Una versión más general de esta función es la siguiente:

```
function a = ental(m,n,a,b)
%ENTAL Matriz entera generada aleatoriamente.
%    ental(m,n) produce una matriz mxn con entradas
%    enteras entre 0 y 9
%    ental(m,n,a,b) produce las entradas de la matriz
entre a y b.
if nargin < 3, a = 0; b = 9; end
a = floor((b-a+1)*rand(m,n))+a;
```

7.2 Archivos de funciones.

Los archivos de funciones hacen que MATLAB tenga capacidad de crecimiento. Se pueden crear funciones específicas para un problema concreto, y a partir de su introducción, tendrán el mismo rango que las demás funciones del sistema. *Las variables en las funciones son locales. Sin embargo, ls versión 5.0 permite declarar una variable como global.*

El ejemplo de la transparencia debe escribirse en el archivo `ental.m` (correspondiente al nombre de la función). **La primera línea** declara el nombre de la función, argumentos de entrada, y argumentos de salida; sin esta línea el archivo sería uno de instrucciones. La instrucción `z = ental(4,5)`, por ejemplo, hará que los números 4 y 5 pasen a las variables m y n en el archivo de función y el resultado se asigna a la variable z. Como las variables en un archivo de función son locales, sus nombres son independientes de los que se encuentren en el espacio de trabajo.

Hagamos notar que el uso de `nargin` ("número de argumentos de entrada"), permite asignar un valor por defecto en caso de que una variable se omita — como a y b en el ejemplo.

Slide 35

función my estad.m

```
function [media, desv] = estad(x)
% ESTAD Media y desviación típica.  Para un vector x,
%    estad(x) da la media y la desviación típica de x.
%    Para una matriz x, estad(x) da dos vectores fila
conteniendo,
%    resp., la media y la desviación típica de
% cada columna.
[m n] = size(x);
if m == 1
   m = n; % caso de un vector fila
end
media = sum(x)/m;
desv = sqrt(sum(x.^2)/m - media.^2)
```

•Una función puede tener también argumentos de salida múltiples. Esto es una vez situado en el archivo de disco `estad.m`, la función de la transparencia; la instrución de MATLAB `[xm, xd] = estad(x)`, por ejemplo, asignará la media y la desviación típica de x a las variables xm y xd, respectivamente.

Cuando se dispone de una función con argumento de salida múltiple, se pueden efectuar asignaciones simples. Por ejemplo, `xm = estad(x)` (no son necesarios los corchetes alrededor de xm) asignará la media de x a xm.

•El símbolo `%` indica que el resto de la línea es un comentario; MATLAB ignorará el resto de la línea. Las primeras líneas de comentario, que documentan el archivo, son accesibles con la instrucción `help`. Así, para que se muestren en pantalla basta escribir `help estad`. Dicha documentación debe incluirse *siempre* en un archivo de función.

•Esta función ilustra algunas de las formas en que MATLAB puede usarse para obtener un código eficiente. Hagamos notar, por ejemplo, que `x.^2` es la matriz de los cuadrados de las entradas de x, que `sum` es una función vectorial, que `sqrt` es una función escalar, y que la división en `sum(x)/m` opera una matriz con un escalar.

Slide 36

máximo común divisor.

```
function a = mcd(a,b)
% MCD Máximo común divisor
%    mcd(a,b) es el máximo común divisor de a y b no
nulos a la vez.
a = round(abs(a)); b = round(abs(b));
if a == 0 & b == 0
   error('El mcd no está definido cuando ambos números
son nulos')
else
   while b ~= 0
      r = rem(a,b);
      a = b; b = r;
   end
end
```

La función de la transparencia, que da el máximo común divisor de dos enteros vía el algoritmo de Euclides, ilustra el uso de un mensaje de error (ver sección siguiente).

Algunas de las funciones de MATLAB son internas mientras que otras se distribuyen como archivos .m. El listado de cualquier archivo de extensión *.m—de MATLAB o del usuario—puede obtenerse con la instrucción de MATLAB `type` *nombre_de_función*. Pruebe con `type eig`, `type vander`, y `type rank`.

Algunas posibilidades más avanzadas se ilustran con la siguiente función. Las entradas de la función—como *tol* en el ejemplo siguiente, puede hacerse opcional mediante el uso de `nargin` ("número de argumentos de entrada"). La variable `nargout` puede usarse de forma similar. Hagamos notar que el hecho de que una relación es un número (1 cuando es cierta; 0 cuando es falsa) es usado, y que, cuando `while` o `if` evalúan una relación, "no cero" significa "cierto" y 0 significa "falso". Finalmente, la función de MATLAB `feval` permite tener como variable de entrada una cadena que dé nombre a otra función.

```
   function [b, pasos] = bisecc(fun, x, tol)
%BISECC Cero de una función de una variable por bisección.
%    •bisecc(fun,x) produce un cero de la función. fun es una cadena
%    conteniendo el nombre de una función real de una variable real;
%    normalmente las funciones están definidas en archivos .m.
%    x es el punto inicial. El valor producido está cerca de un
%    punto donde la función cambia de signo. Por ejemplo,
%    bisecc('sin',3) es pi. Nótense las comillas alrededor de sin.
%    •Un tercer argumento de entrada opcional fija la tolerancia
%    para la precisión relativa del resultado. El valor por defecto
%    es eps. Un argumento de salida opcional produce una matriz
%    con las iteraciones; sus filas son de la forma [c, f(c)].
if nargin < 3, tol = eps; end
traza = (nargout == 2);
if x ~= 0, dx = x/20; else, dx = 1/20; end
a = x - dx; fa = feval(fun,a);
b = x + dx; fb = feval(fun,b);
% Encontrar un cambio de signo.
while (fa > 0) == (fb > 0)
   dx = 2.0*dx;
   a = x - dx; fa = feval(fun,a);
   if (fa > 0) ~= (fb > 0), break, end
   b = x + dx; fb = feval(fun,b);
end
if traza, pasos = [a fa; b fb]; end
% Bucle Principal
while abs(b - a) > 2.0*tol*max(abs(b),1.0)
   c = a + 0.5*(b - a); fc = feval(fun,c);
   if traza, pasos = [pasos; [c fc]]; end
   if (fb > 0) == (fc > 0)
      b = c; fb = fc;
   else
      a = c; fa = fc;
   end    end
```

Slide 37

Cadenas de texto

- `s = 'Esto es una prueba'`
- `disp('Este mensaje se está mostrando aquí')`
- `error('Lo siento, la matriz debe ser simétrica')`
- `iter=input('Introduzca el número de iteraciones:  ')`

8 Cadenas de texto, mensajes de error, input

•Las cadenas de texto se introducen en MATLAB entre comillas simples. Por ejemplo,

```
s = 'Esto es una prueba'
```

asigna la cadena de texto dada a la variable `s`.

•Las cadenas de texto pueden mostrarse con la función `disp`. Por ejemplo:

```
disp('Este mensaje se está mostrando aquí')
```

•Los mensajes de error se muestran mejor con la función `error`

```
error('Lo siento, la matriz debe ser simétrica')
```

ya que ésta hace que la ejecución salga del archivo .m.

•En un archivo .m el usuario puede ser avisado para introducir datos interactivamente con la función `input`. Si MATLAB se encuentra, por ejemplo, con la instrucción:

```
iter = input('Introduzca el número de iteraciones:  ')
```

la cadena entre comillas se muestra y la ejecución se detiene mientras el usuario introduce los datos. Tras pulsar el retorno de carro los datos se asignan a la variable `iter` y continúa la ejecución.

Slide 38

- Mientras se usa MATLAB se necesita usualmente crear o editar un archivo .m y regresar a MATLAB. Sería deseable mantener MATLAB activo mientras se edita un archivo pues, en caso contrario, se perderían todas las variables tras salir.
- Esto puede hacerse fácilmente con el signo !. Si, estando en MATLAB, escribe una instrucción del sistema operativo —como las que se usan para editar, imprimir y copiar un archivo— precedida del signo !, se ejecuta la instrucción sin salir de MATLAB. Si por ejemplo, la instrucción del sistema operativo `edit` accede al editor, la instrucción de MATLAB

```
>> !edit nombre.m
```

le permitirá editar el archivo `nombre.m` usando su editor local. Tras dejar el editor, retornará a MATLAB, justo donde se dejó.

9 Tratamiento de archivos .m

La versión 5.0 dispone de varias herramientas de rastreo. Ver `help dbtype` y las referencias que se dan allí.

Aunque las siguientes instrucciones reflejan las del sistema operativo, evitan el uso de !. Estando en MATLAB, la instrucción `dir` mostrará los contenidos del directorio activo mientras que `what` mostrará sólo los archivos .m en el directorio. Las instrucciones `delete` y `type` sirven para borrar un archivo de disco e imprimirlo en pantalla, respectivamente, y `chdir` para cambiar el directorio de trabajo.

Los archivos .m deben ser accesibles a MATLAB. En la mayoría de los sistemas o instalaciones en red, los archivos .m personales que se almacenan en un subdirectorio del directorio raíz denominado `matlab` serán accessibles para MATLAB desde cualquier directorio en el que se trabaje. Ver la discusión de MATLABPATH, en la Guía del usuario, para más información.

Slide 39

- La función `flops` es un contador de las operaciones realizadas. La instrucción `flops(0)` (no `flops=0`!) inicializa el contador a 0. Por tanto si usamos `flops(0)` inmediatamente antes de ejecutar un algoritmo, la instrucción `flops` situada justo al final nos dará el número de operaciones que se han efectuado en su ejecución.
- La función `clock` da la hora actual aproximada hasta la centésima de segundo (ver `help clock`). Dados dos tiempos `t1` y `t2`, `etime(t2,t1)` proporciona el tiempo transcurrido de `t1` a `t2`.

10 Eficiencia de algoritmos: flops y etime

Dos medidas de la eficiencia de un algoritmo son el número de operaciones realizadas y el tiempo gastado: mediante flops y etime.

Se puede, por ejemplo, medir el tiempo que requiere la resolución de un sistema de ecuaciones dado $Ax = b$ usando eliminación gaussiana como sigue: `tiempo = clock; x = A\b; tiempo = etime(clock,tiempo)` Puede desear comparar éste —y la cuenta `flop`— con los valores que se obtienen usando `x = inv(A)*b;`. Inténtelo.

Hagamos notar que, en máquinas que operan a tiempo compartido, `etime` no es una medida fiable de la eficiencia de un algoritmo ya que la velocidad de ejecución depende de lo ocupada que esté la máquina en un momento determinado.

Slide 40

Formato de salida

Aunque todos los cálculos en MATLAB se efectúan en doble precisión, el formato de la salida en pantalla puede ser controlado con las siguientes instrucciones.

`format short`	coma fija con 4 decimales (el defecto)
`format long`	coma fija con 14 decimales
`format short e`	notación científica con 4 decimales
`format long e`	notación científica con 15 decimales

Una vez que se ordena un formato, se mantiene hasta que se ordena un cambio.

La orden `format compact` evitará la mayor parte de las líneas en blanco, con lo que se puede mostrar más información en pantalla. Es independiente de las demás instrucciones de formato.

Slide 41

Hardcopy

La forma más sencilla de obtener una hardcopy [a] es con la instrucción **diary**. La orden

`diary` *nombre_de_archivo*

hace que todo lo que aparezca a continuación en pantalla (excepto los gráficos) sea escrito en el archivo *nombre_de_archivo* (si se omite el nombre se toma por defecto `diary`) hasta que se ordena `diary off`; la instrucción `diary on` hará que se escriba al final del archivo, etc. Al terminar, se puede editar el archivo como se desee e imprimirlo en el sistema local. Todo se puede hacer sin salir de MATLAB usando el signo ! (ver sección 9).

[a] Una copia por impresora o en un archivo de disco.

Slide 42

Gráficos planos. La instrucción `plot` crea gráficos en el plano XY; si x e y son vectores de la misma longitud, la orden `plot(x,y)` accede a la pantalla gráfica y realiza un gráfico plano de los elementos de x contra los elementos de y.

• Pueden ponerse títulos, comentarios en los ejes o en cualquier otra parte con los siguientes comandos que tienen una cadena como argumento:

title	título del gráfico
xlabel	comentario en el eje x
ylabel	comentario en el eje y
gtext	texto posicionado interactivamente
text	texto posicionado mediante coordenadas

11 Gráficos

MATLAB puede producir gráficos planos y gráficos de malla de superficies tridimensionales.

11.1 Gráficos planos

Para ver algunas de sus posibilidades realice los siguientes ejemplos:

•Por ejemplo, podemos dibujar la gráfica de la función seno sobre el intervalo $[-4, 4]$ con las instrucciones siguientes:

```
x = -4:.01:4; y = sin(x); plot(x,y)
```

donde el vector x es una partición del dominio con paso 0.01, mientras que y es un vector (`sin` es vectorial) con los valores que toma el seno en los nodos de esta partición.

•Como un segundo ejemplo, puede dibujar la gráfica de $y = e^{-x^2}$ sobre el intervalo $[-1.5, 1.5]$ como sigue:

```
x = -1.5:.01:1.5; y = exp(-x.^2); plot(x,y)
```

Hagamos notar que ^ está precedido por un punto para asegurarnos que opera a coordenadas.

Slide 43

- gráficos paramétricos.
- La instrucción `grid` hará un cuadriculado en el gráfico actual.
- Pueden ponerse títulos, comentarios en los ejes o en cualquier otra parte.
- Por defecto, los ejes se autoescalan. Para evitarlo se usa el comando `axis`.
- Existe forma de obtener dibujos múltiples.
- Se pueden evitar los tipos de línea y de punto por defecto.

•Pueden hacerse también gráficos de curvas definidas paramétricamente. Por ejemplo,

```
t=0:.001:2*pi; x=cos(3*t); y=sin(2*t); plot(x,y)
```

•Pueden ponerse títulos, comentarios en los ejes o en cualquier otra parte:

Por ejemplo, la instrucción `title('La función más bella')` proporciona un título al gráfico. El comando `gtext('La mancha')` permite posicionar una cruz en el gráfico con las flechas o el ratón, donde se situará el texto cuando se pulse cualquier tecla.

•Por defecto, los ejes se autoescalan. Para evitarlo se usa el comando `axis`. Si $c = [x_{\min}, x_{\max}, y_{\min}, y_{\max}]$ es un vector con 4 elementos, entonces `axis(c)` establece el escalado de ejes a los límites prescritos. `axis`, por sí mismo congela el escalado actual para gráficos subsecuentes; Escribiendo `axis` de nuevo volvemos al autoescalado. El comando `axis('square')` asegura que se use la misma escala en ambos ejes. En la versión 4.0, `axis` ha sido cambiada significativamente; ver `help axis`.

Slide 44

•Dos formas de obtener dibujos múltiples se ilustran con:

1. `x=0:.01:2*pi;y1=sin(x);y2=sin(2*x);`
   `y3=sin(4*x);plot(x,y1,x,y2,x,y3)`
   y formando una matriz `Y` conteniendo los valores funcionales como columnas `x=0:.01:2*pi; Y=[sin(x)', sin(2*x)', sin(4*x)']; plot(x,Y)`
2. Otra forma es con `hold`. El comando `hold` congela la pantalla gráfica actual de forma que los gráficos posteriores se sobreimponen en ella. Escribiendo `hold` de nuevo se libera el "hold." Los comandos `hold on` y `hold off` también están disponibles en la versión 5.0.

•Se pueden evitar los tipos de línea y de punto por defecto. Por ejemplo,

```
x=0:.01:2*pi; y1=sin(x); y2=sin(2*x); y3=sin(4*x);
plot(x,y1,'--',x,y2,':',x,y3,'+')
```

produce líneas a trazos y de puntos para las dos primeras, mientras que para la tercera se obtiene el símbolo `+` en cada nodo.

Los tipos de líneas y de puntos son:

Tipos de línea: sólido (`-`), a trazos (`--`). puntos (`:`), punto y trazo (`-.`)
Tipos de puntos: punto (`.`), más (`+`), estrella (`*`), círculo (`o`), equis (`x`)
Ver `help plot` para los colores de las líneas y puntos.

El comando `subplot` se usa para dividir la pantalla de forma que puedan verse hasta seis gráficos a la vez. Ver `help subplot`.

Slide 45

Gráficos de malla de superficies tridimensionales.

Para dibujar la gráfica de una función $z = f(x, y)$ sobre un rectángulo, se definen en primer lugar los vectores xx e yy que dan particiones de los lados del rectángulo. Con la función `meshgrid` (mesh domain; el nombre es `meshgrid` en la versión 4.0) se crea una matriz x, en la que cada fila es igual a xx, y de igual forma una matriz y, con todas sus columnas iguales a yy, como sigue:

```
[x,y] = meshgrid(xx,yy);
```

Entonces se computa la matriz z, obtenida evaluando f entrada a entrada sobre las matrices x e y, para aplicarle la función `mesh`.

11.2 Gráficos de malla de superficies tridimensionales.

Los gráficos de malla de superficies tridimensionales se hacen con la función `mesh`. La instrucción `mesh(z)` crea un gráfico tridimensional en perspectiva de la matriz z. La superficie de malla está definida por las coordenadas z de los puntos sobre un cuadriculado rectangular en el plano XY.

•Por ejemplo, pruebe con `mesh(eye(10))`.

• Se puede dibujar la gráfica de $z = e^{-x^2-y^2}$ sobre el cuadrado $[-2, 2] \times [-2, 2]$ como sigue (inténtelo):

```
xx = -2:.1:2;
yy = xx;
[x,y] = meshgrid(xx,yy);
z = exp(-x.^2 - y.^2);
mesh(z)
```

Se podría, desde luego, cambiar las tres primeras líneas en lo anterior por

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

Para más detalles sobre `mesh`, ver la Guía del usuario.

En la versión 5.0, se han ampliado considerablemente las posibilidades gráficas respecto a las superficies tridimensionales. Consulte la ayuda para `plot3, mesh`, y `surf`.