Álgebra Lineal – Tutorial básico de MATLAB

MATLAB es un programa interactivo para cómputos numéricos y visualización de datos. Hay muchas librerías disponibles que extienden las funciones básicas de MATLAB a diferentes áreas de aplicación. Este es un tutorial presenta de manera muy concisa los primeros pasos para empezar a usar Matlab. Se recomienda seguir el tutorial al tiempo que ejecuta MATLAB en otra ventana. De esta manera usted podrá ensayar lo que va a prendiendo. Usted debería poder copiar y pegar cada instrucción que aparece aquí para obtener el mismo resultado en MATLAB.

1. Vectores

Empecemos por crear algo sencillo, como un vector. Ingrese cada entrada del vector separadas por espacio y entre corchetes.

```
[1 2 3]
```

```
ans =
     1     2     3
```

MATLAB crea el vector y lo almacena en una variable temporal llamada `ans`. Para almacenarlo en una variable permanente, basta darle un nombre, por ejemplo `v`, igualando `v` al vector, así

```
v = [1 2 3 4 5]
```

```
v =
     1     2     3     4     5
```

Es posible crear vectores con cierta estructura. Por ejemplo supongamos que queremos un vector con los números pares entre el 0 y el 10:

```
t = 0:2:10
```

```
t =
     0     2     4     6     8    10
```

Manipular vectores es también muy fácil. Supongamos que queremos sumarle 3 a cada entrada del vector `v` que creamos arriba. Entonces basta ejecutar:

```
v+3
```

```
ans =
     4     5     6     7     8
```

De nuevo el resultado es almacenado en la variable temporal `ans`. Si queremos almacenarlo para usarlo más tarde podemos ejecutar

```
w = v + 3
```

```
w =
     4     5     6     7     8
```

Si queremos sumar dos vectores, por ejemplo `v` y `w`, podemos ejecutar:

```
suma = v + w
```

```
suma =
     5     7     9    11    13
```

El producto punto se puede calcular con la función `dot`. Los vectores se escriben entre paréntesis separados por comas, así:

```
dot(v,w)
```

```
ans =
100
```

Otras funciones que te pueden interesar son `norm` que calcula la norma (longitud) de un vector y `cross` que calcula el producto cruz de dos vectores.

2. Funciones

MATLAB incluye muchas funciones estándar. Por ejemplo las funciones matemáticas `sin, cos, ... log, exp, sqrt`, así como muchas otras más especializadas. También incluye constantes usadas comúnmente como π, o i (la raíz cuadrada de -1). Estas se pueden usar directamente

```
sin(pi/4)
```

```
ans =
    0.7071
```

También es posible aplicar muchas de estas funciones a vectores, por ejemplo

```
v = pi/2 * [0:3]
cos(v)
```

```
v =
         0    1.5708    3.1416    4.7124
ans =
    1.0000    0.0000   -1.0000   -0.0000
```

Para determinar como se usa cualquier función utilice la ayuda de MATLAB.
MATLAB también te permite crear tus propias funciones utilizando el comando `function`.

3. Formato

La instrucción `format` controla el formato de salida de los valores numéricos presentados pantalla. Hay tres posibilidades para esta instrucción

- `format short` (muestra 5 dígitos decimales).
- `format long` (muestra 15 dígitos decimales).
- `format rat` (muestra un cociente de enteros).

Por ejemplo si quieres calcular $1 + (2/3)$ en los diferentes formatos obtienes

```
format short
1+(2/3)
format long
1+(2/3)
format rat
1+(2/3)
```

```
ans =
    1.6667
ans =
   1.66666666666667
ans =
     5/3
```

El cambio de formato afecta todas las instrucciones que ejecutes después.

4. Matrices

Las matrices en MATLAB se crean igual que los vectores, excepto que cada fila se separa con un punto y coma (;). Por ejemplo

```
B = [1 2 3 4; 5 6 7 8; 9 10 11 12]
```

```
B =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

Las matrices se pueden manipular de muchas maneras. Puedes sumar o restar matrices

```
[1 1; 0 1] + [-1 1; 1 -1]
```

```
ans =
     0     2
     1     0
```

Puedes hallar la transpuesta colocando una comilla sencilla (') después del nombre de la matriz:

```
C = B'
```

```
C =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
```

También puedes multiplicar las matrices `B` y `C` así:

```
D = B * C
```

```
D =
    30    70   110
    70   174   278
   110   278   446
```

Cabe anotar que la multiplicación solo funciona si las dimensiones de las matrices son compatibles. Si inentas por ejemplo multiplicar B con B obtiene un error.

```
B * B
```

```
>> B*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

También es posible multiplicar dos matrices componente a componente (una especie de producto punto) así:

```
E = [1 2; 3 4]
F = [2 3; 4 5]
G = E .* F
```

```
E =
     1     2
     3     4
F =
     2     3
     4     5
G =
     2     6
    12    20
```

Si tienes una matriz cuadrada, puedes multiplicarla por sigo misma tantas veces como quieras elevandola a la potencia deseada.

```
E^3
```

```
ans =

    37    54
    81   118
```

Si en cambio lo que quieres es elevar cada entrada de la matriz a una potencia dada, puedes hacerlo así:

```
E.^3
```

```
ans =

     1     8
    27    64
```

También es posible acceder a las partes que forman una matriz. Para extraer la entrada i, j colocamos `(i,j)` al frente del nombre de la matriz, por ejemplo

```
H=[10 20 30; 40 50 60]
H(2,3)
```

```
H =
    10    20    30
    40    50    60
ans =
    60
```

Para extraer una fila completa utilizamos dos puntos (:) para indicar que queremos toda la fila

```
H(2,:)
```

```
ans =
    40    50    60
```

Y lo propio para extraer una columna

```
H(:,3)
```

```
ans =
    30
    60
```

También puedes construir una matriz a partir de sus partes. Se pueden juntar columnas

```
u = [1;2;3]
v = [4;5;6]
[u v]
```

```
u =

     1
     2
     3
v =
     4
     5
     6
ans =
     1     4
     2     5
     3     6
```

y también juntar filas

```
u = [1 2 3]
v = [4 5 6]
[u; v]
```

```
u =
     1     2     3
v =
     4     5     6
ans =
     1     2     3
     4     5     6
```

MATLAB también tiene varias instrucciones que permiten construir matrices usadas comunmente como la matriz identidad (`eye(n)`), la matriz cero (`zeros(m,n)`), la matriz de unos (`ones(m,n)`), o una matriz diagonal (`diag([...])`).

```
eye(4)
zeros(2,3)
ones(3,2)
diag([1 -1 2])
```

```
ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
ans =
```

```
     0     0     0
     0     0     0
ans =
     1     1
     1     1
     1     1
ans =
     1     0     0
     0    -1     0
     0     0     2
```

La función (`rand(m,n)`) genera una matriz $m \times n$ cuyas entradas son números aleatorios entre cero y uno. Intenta ejecutar la siguiente instrucción varias veces y verás que cada vez obtienes una matriz diferente.

```
rand(2,2)
```

```
ans =
    0.9501    0.6068
    0.2311    0.4860
```

5. Algebra Lineal en MATLAB

MATLAB tiene extensas funciones relacionadas con álgebra lineal. Por ejemplo, supongamos que quieres resolver el sistema de ecuaciones lineales

$$x - y = 2$$
$$x + y = 3$$

Una manera de hacerlo es hallar la forma escalonada reducida de la matriz aumentada del sistema

```
A = [1 -1; 1 1]
b = [2; 3]
C = [A b]
R = rref(C)
```

```
A =
     1    -1
     1     1
b =
     2
     3
C =
     1    -1     2
     1     1     3
R =
    1.0000         0    2.5000
         0    1.0000    0.5000
```

Otra manera es utilizar el operador (\) que encuentra UNA solución del sistema

```
A\b
```

```
sln =
    2.5000
    0.5000
```

Otra manera más es multiplicando por la inversa de la matriz. La inversa la puedes encontrar con el comando `inv`.

```
D = inv(A)
D*b
```

```
D =
    0.5000    0.5000
   -0.5000    0.5000
ans =
    2.5000
    0.5000
```

También puedes encontrar el determinante de una matriz

```
det(A)
```

```
ans =
     2
```

o los valores propios de una matriz

```
E = [1 2; 3 4]
eig(E)
```

```
ans =
   -0.3723
    5.3723
```

Incluso hay una función para hallar el polinomio característico de una matriz. Recordemos que si A es una matriz $n \times n$, entonces el polinomio característico de A es el polinomio de grado n dado por la ecuación $p(A) = \det(A - \lambda I_n) = a_n\lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0$. Cuando n es **par** la función `poly` crea un vector con los coeficientes $a_n, a_{n-1}, \ldots a_0$ del polinomio característico. Cuando n es **impar** la función `poly` crea un vector con los coeficientes $-a_n, -a_{n-1}, \ldots, -a_0$. En otras palabras, cuando la dimensión de la matriz A es impar la función `poly` produce los negativos de los coeficientes del polinomio característico. Por ejemplo si E es la matriz 2×2 definida arriba obtenemos

```
p = poly(E)
```

```
p =
    1.0000   -5.0000   -2.0000.
```

Esto significa que el polinomio característico de E es $p(\lambda) = \lambda^2 - 5\lambda - 2$. Por otro lado, si J es la siguiente matriz 3×3

```
J = [1 2 0; 3 4 -1; 1 0 -1]
```

obtenemos

```
q= poly(E)
```

```
q =
    1.0000   -4.0000   -7.0000    0.0000.
```

Esto significa que el polinomio característico de J es $p(\lambda) = -\lambda^3 + 4\lambda^2 + 7\lambda$.

Recuerda que los valores propios de una matriz son las raíces de su polinomio característico, para matriz E se obtiene:

```
roots(p)
```

```
ans =
    5.3723
   -0.3723
```

Otro comando bastante útil es `null`. Este permite hallar una base del espacio nulo de una matriz

```
X = [1 -2; -2 4]
null(X)
```

```
ans =
    0.8944
    0.4472
```

En ocaciones es preferible obtener una base con coeficientes racionales, para esto incluimos el argumento `'r'` al comando `null`

```
null(X, 'r')
```

```
ans =
    2
    1
```

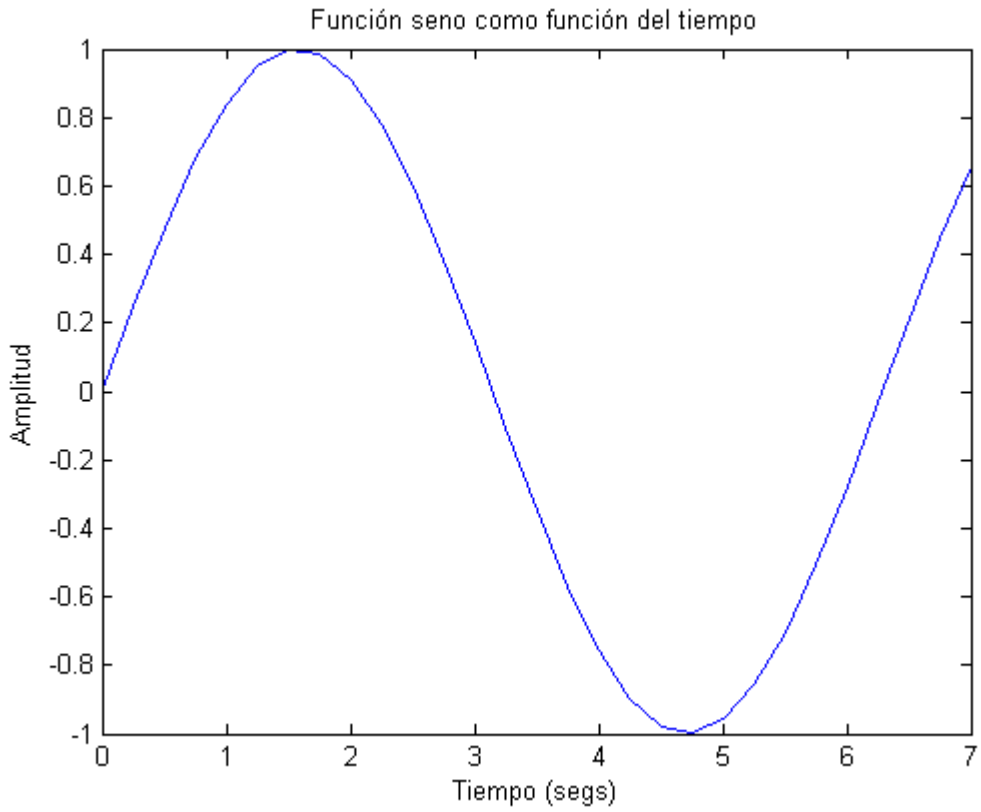
6. Gráficas

Es fácil generar gráficas en MATLAB utilizando el comando `plot`. Este comando no grafica directamente funciones, sino que grafica puntos en un plano cartesiano. De modo que para graficar una función, debes entregarle los puntos que componen la función.
Supongamos que quieres graficar la función $sen(x)$ en el intervalo $[0, 7]$. Primero debes crear un vector con los valores de x que quiere incluir. Luego debes calcular el seno en cada uno de los valores de x, y finalmente utilizar `plot` para graficar los puntos.

```
x = 0:0.25:7;
y = sin(x);
plot(x,y)
```

Note que colocamos un punto y coma al final de las primeras dos instrucciones. Esto se hace para que MATLAB no imprima el resultado en pantalla. Note que la primera instrucción crea un vector de 28 entradas, desde cero hasta 7 separadas por 0,25. La segunda instrucción calcula el seno de cada una de las entradas del vector `x`.
Es posible agregar títulos y anotaciones a las gráficas utilizando las funciones `title`, `xlabel`, y `ylabel` después de haber ejecutado el comando `plot`.

```
title('Función seno como función del tiempo')
xlabel('Tiempo (segs)')
ylabel('Amplitud')
```



La función `plot` tiene una cantidad de otras características que se pueden explorar en la ayuda o en multitud de tutoriales en internet.

7. Polinomios

En matlab, un polinomio es representado por un vector. Para crear un polinomio en MATLAB, simplemente ingresa cada coeficiente del polinomio en orden descendiente en un vector. Por ejemplo, el polinomio $x^4 + 3x^3 - 15x^2 - 2x + 9$ se ingresa en MATLAB como:

```
x = [1 3 -15 -2 9]
```

```
x =
     1     3   -15    -2     9
```

MATLAB interpreta un vector de de $n + 1$ componentes como un polinomio de grado n. Entonces, si a tu polinomio le falta algún coeficiente, debes ingresar ceros en las entradas apropiadas del vector. Por ejemplo, $x^4 + 1$ se representa en MATLAB como el vector:

```
y = [1 0 0 0 1]
```

```
y =
     1     0     0     0     1
```

Puedes evaluar un polinomio utilizando la función `polyval`. Por ejemplo, para evaluar el polinomio definido arriba en s=2 ejecutamos,

```
z = polyval([1 0 0 0 1],2)
```

```
z =
    17
```

También puedes extraer las raíces de un polinomio usando la función `roots`. Las raices del polinomio $x^4 + 3x^3 - 15x^2 - 2x + 9$ se encuentran ejecutando

```
roots([1 3 -15 -2 9])
```

```
ans =
   -5.5745
    2.5836
   -0.7951
    0.7860
```

Para multiplicar dos polinomios se utiliza la función `conv` que encuentra la convolución de sus coeficientes.

```
x = [1 2];
y = [1 4 8];
z = conv(x,y)
```

```
z =
     1     6    16    16
```

La función `deconv` divide un polinomio enre otro y devuelve el residuo y el cociente de la división.

```
[xx, R] = deconv(z,y)
```

```
xx =
     1     2
R =
     0     0     0     0
```

8. Archivos .m

Es posible guardar un listado de instrucciones MATLAB en un archivo con extensión .m para ejecutar más tarde. Bajo Windows MATLAB tiene un editor de archivos .m.

9. Alternativa a Matlab Gratuita – Octave

MATLAB es un excelente programa para cómputos numéricos pero es costoso y no es el único. Existen alternativas gratuitas también muy poderosas. Octave es un software libre que imita a Matlab de manera muy cercana. Octave se puede obtener libremente en internet. Aunque es más fácil usarlo en Linux, también es posible instalarlo en Windows o Mac. Para usuarios de Windows, se recomienda por ejemplo descargar el instalador de Octave de `http://mxeoctave.osuv.de/`. Una vez instalado, todos los comandos usados en el Tutorial de Matlab se pueden usar directamente en Octave. En Windows 8 hay algunos problemas para producir gráficas.