Programación con Matlab

Práctica 1a: Introduciendo valores en el entorno de Matlab

Prof. Guilmer Gonźalez

El curso está dirigido a los estudiantes de la Facultad de Ciencias de la UNAM, en su segundo semestre de la carrera. El objetivo es aprovechar las ventajas de este *entorno de programación* que les permita implementar sus trabajos de clase. El curso consta de varias prácticas, esa será la estrategia para aprender Matlab.

Requisitos:

1) Buena disposición, paciencia y curiosidad.

2) Conocimientos elementales de vectores, sistemas de ecuaciones lineales.

3) La experiencia en temas de programación es útil, más no indispensable.

Matlab es un interprete de instrucciones, mismas que deben ser introducidas en la línea de comandos `>>` que aparece en el entorno de Matlab. La estructura básica de Matlab son los arreglos de datos o vectores. Para introducir un vector o matriz, use corchetes cuadrados para indicar cuando inicia y cuando finaliza los datos; por ejemplo, a introducir

```
>>y = [0 1 2 3 4]
```

observe que se despliega en pantalla el contenido del vector `y`

```
y =

     0     1     2     3     4
```

Para introducir un vector con más de un renglón o bien los elementos de una matriz por renglones, debe usar el caracter `;`, como separador entre la información de los renglones; por ejemplo tecleando

```
>>z = [0; 1; 2; 3; 4]
```

se tiene como resultado:

```
z =

     0
     1
     2
     3
     4
```

Identificando la forma de introducir cada renglón, podemo ahora jugar con matrices. Para introducir la matriz

$$A = \begin{pmatrix} 0 & 2 \\ 3 & 7 \\ 12 & 8 \end{pmatrix}$$

use

```
>>A = [0 2;3 7; 12 8]
```

Observe en pantalla la forma en que se muestra la matriz.

Si cuenta con un vector en memoria y necesita trabajar con su matriz transpuesta, matlab tiene un identificador para ello, la comilla simple. Teclee y observe

```
>> w=z'

w =

     0     1     2     3     4
```

Ahora, haga

```
>> B=A'
```

Una de las ventajas de Matlab es poder jugar con los vectores, construirlos muy fácilmente si estos provienen de una colección de datos simple. Por ejemplo, si quiere generar 10 números del 1 al 10, puede hacerlo en la forma

```
>> t = 1:10

t =

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

Aquí el caracter : es un separado entre el índice incial y el final. También puede generar números con cierto espaciamiento entre ellos. Generemos números igualmente espaciados entre 0 y 1, con espaciamiento .15

```
>> alpha = 0:0.15:1
```

cuántos números se obtuvieron de esta forma? Es fácil saber si los cuenta, si hace la cuenta adecuada será más rápido en saber.
Nota: Matlab asume valores predeterminados y acciones predeterminadas, si en una colección de datos generadas de esta forma no expecifica el espaciamiento, Matlab asume que este es 1.

Pruebe las siguientes secuencias, y observe pero sobre todo, idenfique si son válidas y su caso, de antemano asuma su resultado:

a) `>> h = 0:.01:1,`

b) `>>  d=2:4.3,`

b) `>>  ind=3:0,`

d) `>>  iter=[1 3 5 7 8 9 11],`

Accediendo a los datos de una matriz o vector

Nuestros problemas numéricos requieren acceder a los datos que se van generando, ya sea para modificarlos o para observar el valor que aproximamos digamos, en el segundo valor propio más pequeño de una matriz.

Sigamos trabajando con los vectores que hemos definido en nuestro entorno de Matlab. Podemos especificar la posición dentro del vector del valor a observar, por ejemplo:

```
>> y(3)

ans =

     2
```

en el caso de matrices, debemos especificar la posición renglón-columna del valor a acceder, por ejemplo

```
>> A(3,2)

ans =

     8
```

Pero debemos tener cuidado de no acceder a posiciones no existes, Matlab produce un error y si estamos dentro de un programa, este aborta señalando el error, por ejemplo:

```
>> A(3,3)
???  Index exceeds matrix dimensions.
```

Por supuesto, en esta programación podemos cambiar cada uno de los elementos o bien, solo modificar uno, por ejemplo

```
>> y(3)=y(3)+2

y =

     0     1     4     3     4
```

Ha observado la diferencia con respecto al arreglo original?

Pruebe y observe que produce:

```
>> A(3,2) = A(3,2) - 3
```

Lo interesante de programar en lenguajes de 4ta generación, es que podemos acceder a parte de la información de Matrices y vectores y jugar con ellos, por ejemplo si deseamos obtener los dos primeros elementos sobre el segundo renglón de la matriz `B`, podemos teclear

```
>> B(2,1:2)
```

si queremos todo el segundo reglón, podemos hacer uso de caracter especial `:` en la forma

```
>> B(2,:)
```

Este caracter manejado adecuadamente nos permitirá manipular una matriz por bloques, cambiar su valor, acceder a el.

Matlab es versátil, permite obtener información de los vectores de manera transparente al usuario. Podemos estar interesados en obtener los 2 últimos elementos del segundo reglón de B, para ello podemos teclear

```
>> B(2,end-1:end)
```

Observe en pantalla cuántas variables contamos en el entorno de Matlab así como el tamaño de los elementos usando el comando `whos`:

```
>> whos
  Name        Size          Bytes  Class

  A           3x2              48  double array
  B           2x3              48  double array
  alpha       1x7              56  double array
  ans         1x1               8  double array
  t           1x10             80  double array
  w           1x5              40  double array
  y           1x5              40  double array
  z           5x1              40  double array

Grand total is 45 elements using 360 bytes
```

Operaciones elementales

Recordemos, en Matlab las variables son vectores, por lo que podemos "jugar" con ellos, añadiendo, restando multiplicando o dividiendo siguiendo las reglas del Álgebra Lineal; esto es, no podemos dividir por cero desde luego, ni multiplicar una matrix de 2×3 por una de 4×4. Si se atreve a hacer, Matlab le dará un sape.

Si multiplicamos un escalar por un vector o matriz por un escalar, obtendremos una matriz cuyas entradas son el producto de los valores correspondientes con el escalar, por ejemplo:

```
>> C = 2*[1 2; 3 4],
C =

     2     4
     6     8
```

en el caso de vectores

```
>> v=2*[1 2 3 4]

v =

     2     4     6     8
```

Cuando añadimos un escalar a un vector, Matlab nos genera como salida un vector, donde cada entrada es añadida por ese escalar

```
>> d = 2 + [0:5]
d =

     2     3     4     5     6     7
```

y desde luego, podemos añadir y multiplicar escalares por un vector en una sola operación

```
>> e = 4 + 3.2*[2:4]

e =

   10.4000   13.6000   16.8000
```

Si tenemos dos matrices de la misma dimención

```
>>A=[ 1  2  3;  6  5  4;  1  2  4];
>>B=[ 1 -2 -3; -6  5 -4; -1 -2  4];
```

podemos añadirlas

```
>> A+B

ans =

     2     0     0
     0    10     0
     0     0     8
```

o hacer la diferencia entre ellas

```
>> A-B

ans =

     0     4     6
    12     0     8
     2     4     0
```

y hacer la multiplicación entre matrices

```
>> A*B

ans =

   -14     2     1
   -28     5   -22
   -15     0     5
```

Existen otras operaciones entre elementos de vectores y matrices que podemos hacer, y esas las conecemos como las operaciones "puntito" `.*` `.^` `./` básicamente. Por ejemplo, si tenemos dos vectores de la misma dimensión y hacemos la operación

```
>> [1 2 3 4 5].*[2 0 3 2 1]

ans =

     2     0     9     8     5
```

La salida es un nuevo vector formado al multiplicar cada estrada del primero por la correspondiente del segundo. Lo propio ocurre al usar `./\`. Esta característica de este lenguaje de programación nos permite "jugar" con los elementos de arreglos de manera compacta, sencilla y rápida. Por ejemplo si necesitamos evaluar una función $f(x) = x^2\cos(x) + 1/x$, en una colección de puntos `a`, en matlab tecleamos

```
>> sal = a.^2.*cos(a) + 1./a
```

Con esto podemos evaluar funciones o hacer cálculos rápidamente si logramos observarlos como una colección de arreglos y aplicando las operaciones "puntito" podemos evaluar todo un vector siguiendo adecuadamente las reglas.

Operando con matrices

En Matlab muchas operaciones son predeterminadas, recordemos que Matlab inicia como un entorno de programación basado en las bibliotecas linpack y eispack, podemos resolver en línea (en el entorno de Matlab) sistemas de ecuaciones y lograr su inversa usando un par de instrucciones. Por ejemplo, contando con una matriz asociada un sistema de 3×3 y el lado derecho:

```
>> A=[1 2 3;6 5 4;1 2 4], b=[1 ; 13 ; 0],
```

si usamos la expresión `>> A\b` Matlab nos devuelve la solución del sistema $Ax = b$, realice esta operación en el entorno de programación.

Si usamos `det(A)` tenedremos el determinante de la matriz. Si usamos `inv(A)`, obtendremos la inversa (la inversa generalizada de A si esta no es cuadrada o es singular (revise la teoría existente, o busque en google).

La primera biblioteca que se programó eficientemente fue eispack, necesitabamos resolver problemas de valores y vectores propios antes de reolver

sistemas de ecuaciones, en ese entonces la guerra movia la industria incluso la tecnología. En Matlab, podemos calcular los valores propios y vectores propios de una matriz usando la expresión

```
[V, D]=eig(A)
```

introduzca esa instrucción en el entorno y observe que `V` es una matriz ortonormal, estos es, las columnas son ortogonales y de norma 1, y `D` es una matriz diagonal que contiene a los correspondientes valores propios. Para obtener la transpuesta de una matriz use `'`; por ejemplo `u'` es la transpuesta del vector `u`. Verifique `V` es una matriz ortogonal (calcule `V'*V`).
cada elemento del primer vector o matriz se multiplica con el correspondiente elemento o entrada del vector o matriz formando un nuevo arreglo conteniendo el resultado.

Matlab cuenta con muchos comandos para generar matrices o manipular las existentes, veremos algunos ejemplos rápidamente, pero el interesado puede teclear

```
>> help elmat                o
>> help matfun
```

para que obtenga una lista de funciones y comandos para operar con matrices.

Podemos crear una matriz de ceros introduciendo `A = zeros(3,4)` o bien una matriz llena de unos con el comando `B=ones(5,3)`. Desde luego, podemos generar un vector con estas instrucciones, basta con indicar si será columna o renglón. Haga una práctica: genere vectores columna.

En Matlab podemos construir matrices identidad bajo la instrucción

```
I = eye(5)

I =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Como se observa es suficiente indicar la dimensión de la matriz, ya que todas las matrices identidad son cuadradas.

En Matlab como se comentó al inicio, podemos identificar o extraer información de la colección de datos sea por columnas o por filas de una manera muy sencilla; por ejemplo, si deseamos obtener el tercer renglón

```
>> I(3,:)
ans =

     0     0     1     0     0
```

o la columna `I(:,4)`. De igualmanera podemos elegir una submatriz o usar sus elementos indicando la secuencia de columnas y renglones, ejemplo, teclee `I(1:3,2:4)` y observe con la matriz original.

Por último, qué resultado obtiene al hacer las siguientes operaciones:

a) `>> E = eye(4); E(2,1) - 3,`

b) `>>  B = eye(3); P=E([2 1 3,:),`