Introducción a MATLAB

Manejo de polinomios

Table of Contents

A lo largo de nuestra vida académica hemos ido acumulando experiencia con **polinomios**: expresiones algebraicas en las que intervienen números (coeficientes) y letras (variables) que están relacionadas mediante operaciones como sumas, multiplicaciones y/o potencias. Los polinomios se definen en MATLAB mediante vectores fila con los coeficientes del polinomio en cuestión en orden de potencias decrecientes. Por ejemplo, si queremos definir el polinomio $p(x) = x^3 - 2x - 5$ se procedería como sigue:

```
% p = x^3 - 2*x - 5
% Polinomio de tercer grado, con lo que:
% Coeficiente que acompaña a x^3: 1
% Coeficiente que acompaña a x^2: 0
% Coeficiente que acompaña a x^1: -2
% Término independiente: -5
% Resultando en la definición:
p = [1 0 -2 -5]

p =

     1     0    -2    -5
```

De manera general, el polinomio $p(x) = p_2x^2 + p_1x + p_0$ se representa en Matlab como:

```
p = [p2 p1 p0]
```

Una vez presentados nuestros protagonistas, ¡trabajemos un poco con ellos!

1. Trabajando con polinomios

1.1 Evaluando polinomios

Para evaluar un polinomio en un cierto punto MATLAB nos ofrece el comando **`polyval`**. Por ejemplo, $p(4)$ se puede calular como:

```
res = polyval(p,4)

res =
```

```
51
```

Alternativamente, también se puede evaluar un polinomio desde el punto de vista matricial usando **polyvalm**.

Por ejemplo, el polinomio $p(x) = x^3 - 2x - 5$ se convertiría en la expresión matricial $p(X) = X^3 - 2X - 5I$, dónde X es una matriz cuadrada y I es la matriz identidad. Por ejemplo:

```
p = [1 0 -2 -6];
X = [3 -2; 0 3];
Y = polyvalm(p,X)

Y =

    15   -50
     0    15
```

Tarea 1: Definir el polinomio $q(x) = -2x^5 + 3x^3 - 2x + 5$ y evaluarlo en $x = 2$.

```
% Tu código aquí
q = [-2 0 3 0 -2 5]
```

```
q = 1×6
    -2     0     3     0    -2     5
```

```
x = 2
```

```
x = 2
```

```
polyval(q,x)
```

```
ans = -39
```

```

```

Resultado esperado: `q_x = -39`

1.2 Obteniendo las raíces de un polinomio

Las raíces de un polinomio pueden calcularse empleando el comando **roots**. Por ejemplo:

```
r = roots(p)

r =

   2.0946 + 0.0000i
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
```

Tarea 2: Calcula las raíces del polinomio $q(x)$ previamente definido.

```
% Tu código aquí
roots(q)
```

```
ans = 5×1 complex
  -1.1927 + 0.6181i
  -1.1927 - 0.6181i
   1.3854 + 0.0000i
   0.5000 + 0.8660i
   0.5000 - 0.8660i
```

1.3 Multiplicando polinomios

Si estamos trabajando con dos polinomios, estos pueden multiplicarse empleando el comando **conv**. Por ejemplo, si se desea multiplicar el ya conocido polinomio $p(x) = x^3 - 2x - 5$ por $p_2(x) = 4x^3 + 5x + 6$

```
p2 = [4 5 6]

p2 =

     4     5     6

p3 = conv(p,p2)

p3 =

     4     5    -2   -30   -37   -30
```

Lo que da como resultado el polinomio $p_3(x) = 4x^5 + 5x^4 - 2x^3 - 30x^2 - 37x - 30$.

Este comando también es útil para construir un polinomio a partir de su factorización. Por ejemplo si $q(x) = (x-2)(x+3)(x+5i)(x-5i)$ podemos obtener el polinomio del que proviene dicha factorización como sigue:

```
q = conv(conv(conv([1 -2],[1 3]),[1 5i]),[1 -5i])

q =

     1     1    19    25  -150
```

Lo que da como resutlado $q(x) = x^4 + x^3 + 19x^2 + 25x - 150$.

Nota: Nótese que la convolución de dos vectores es equivalente a la multiplicación si estos representan coeficientes polinomiales.

<u>**Tarea 3:**</u> Obtén el polinomio $q(x)$ resultante de la factorización $q(x) = (x+3)(x+2+3i)(x+2-3i)$. Comprueba que el polinomio obtenido es el correcto calulando sus raíces y comparándolas con la factorización inicial.

```
% Tu código aquí
 q = conv(conv([1 3],[1 2+3*i]),[1 2-3*i])
```

```
q = 1×4
     1     7    25    39
```

```
roots(q)
```

```
ans = 3×1 complex
  -2.0000 + 3.0000i
  -2.0000 - 3.0000i
  -3.0000 + 0.0000i
```

```
roots([1 2+3*i])
```

```
ans = -2.0000 - 3.0000i
```

```
roots([1 2-3*i])
```

```
ans = -2.0000 + 3.0000i
```

1.4 Obteniéndo el polinomio característico de una matriz

El polinomio característico de una matriz se calcula como $p_A(\lambda) = \mathrm{def}(A - \lambda I)$. En álgebra, este polinomio característico es una herramienta ampliamente utilizada ya que provee gran cantidad de información sobre la matriz, como por ejemplo los valores propios, el determinante y su traza.

En MATLAB el polinomio característico de una matriz puede obtenerse con el comando **poly**. Por ejemplo:

```
A =

     2     3     4
     4     5     2
     1     8     6

pc = poly(A)

pc =

    1.0000  -13.0000   20.0000  -70.0000
```

Tarea 4: Calcular, a partir de la matriz `A=[2 4 3; -1 3 2; 8 3 -9]`, un vector p constituido por los coeficientes del polinomio característico de la matriz `A`.

```
% Tu código aquí
 A=[2 4 3; -1 3 2; 8 3 -9];
 p = poly(A)
```

```
p = 1×4
    1.0000    4.0000  -65.0000  119.0000
```

Resultado esperado:

```
pc = 1×4
     1.0000    4.0000  -65.0000  119.0000
```

Tarea 5: El comando **poly** tiene un segundo uso,interesante. Para ilustrarlo, calcular, a partir del vector `p = [1 0 -2 -5]`, un nuevo vector `r` constituido por las raíces de dicho polinomio.

```
% Tu código aquí
p = [1 0 -2 -5];
r= roots(p)
```

```
r = 3×1 complex
   2.0946 + 0.0000i
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
```

poly permite obtener el vector de coeficientes de un polinomio a partir de un vector que contiene sus raíces. Por ejemplo, empleando las raíces de nuestro polinomio de ejemplo `p`:

```
p4 = poly(r)

p4 =

    1.0000   -0.0000   -2.0000   -5.0000
```

De este modo, los comandos **poly** y **roots** pueden verse como inversos: con **roots** calculo raíces que puedo convertir en polinomio con **poly**, y viceversa.

<u>**Tarea 6:**</u> Calcular, a partir del vector `r`, un nuevo vector `p4` constituido por los coeficientes de un polinomio cuyas raíces son los elementos del vector `r`. Comprobar que el vector `p4` obtenido coincide con el vector `p` calculado anteriormente.

```
% Tu código aquí
p4 = poly(r)
```

```
p4 = 1×4
    1.0000   -0.0000   -2.0000   -5.0000
```

[BOMBA] El comando salvador: help

Si tenemos dudas sobre el uso de algún comando o función, podemos emplear el comando **help** para obtener ayuda sobre el mismo. Por ejemplo, ejecuta la siguiente celda de código para revisar la documentación relativa al comando **poly**:

```
help poly
```

Si seguimos teniendo dudas, vistar la documentación del comando puede ser útil.

1.5 División, cociente y resto

El cociente y el resto de la división de dos polinomios puede obtenerse mediante el comando **deconv**. Por ejemplo:

```
[q,r] = deconv(p,p2)

q =

    0.2500   -0.3125


r =

         0         0   -1.9375   -3.1250
```

Tarea 7: Calcular, a partir de los vectores `p = [1 0 -2 -5]` y `p2 = [1 4 5 6]`, unos nuevos vectores `coc` y `res` constituidos, respectivamente, por los coeficientes de los polinomios cociente y resto de la división del polinomio cuyos coeficientes son los términos del vector `p2` entre el polinomio cuyos coeficientes son los términos del vector `p`.

¿Un poco de ayuda sobre **deconv**?

```
help deconv
```

```
deconv - Least-squares deconvolution and polynomial division
   This MATLAB function deconvolves a vector h out of a vector y using
   polynomial long division, and returns the quotient x and remainder r
   such that y = conv(x,h) + r.

   Polynomial Long Division
     [x,r] = deconv(y,h)

   Least-Squares Deconvolution
     [x,r] = deconv(y,h,shape)
     [x,r] = deconv(___,Name=Value)

   Input Arguments
     y - Input signal to be deconvolved
       row or column vector
     h - Impulse response or filter used for deconvolution
       row or column vector
     shape - Subsection of convolved signal
       "full" (default) | "same" | "valid"

   Name-Value Arguments
     Method - Deconvolution method
       "long-division" (default) | "least-squares"
     RegularizationFactor - Tikhonov regularization factor
       0 (default) | real number

   Output Arguments
     x - Deconvolved signal or quotient from division
       row or column vector
     r - Residual signal or remainder from division
       row or column vector

   Examples
     Polynomial Division
     Least-Squares Deconvolution of Fully Convolved Signal
     Least-Squares Deconvolution of Central Part of Convolved Signal
     Least-Squares Deconvolution Problem with Infinite Solutions
     Specify Regularization Factor for Noisy Signal
```

```
    See also conv, residue

    Introduced in MATLAB before R2006a
    Documentation for deconv
```

```
% Tu código aquí
p = [1 0 -2 -5];
p2 = [1 4 5 6];
[coc,res] = deconv(p2,p)
```

```
coc = 1
res = 1×4
     0     4     7    11
```

Comprueba que el resultado es correcto calculando el producto del cociente por p y sumandole el resto.

```
% Tu código aquí
p2 = (coc * p) + res
```

```
p2 = 1×4
     1     4     5     6
```

Resultado esperado:

```
p2 = 1×4
     1     4     5     6
```