

## 5.2 Polinomios

En las ciencias y la ingeniería, los **polinomios** son uno de los modelos más comunes para la representación del comportamiento de sistemas físicos.

En general, un polinomio es una relación de una serie de potencias; expresada en forma de una función en una sola variable y que usualmente se denota como:

$$f(x) = c_1 x^n + c_2 x^{n-1} + c_3 x^{n-2} + \dots + c_{n-1} x^2 + c_n x + c_{n+1}$$

O en forma factorizada:

$$f(x) = c_1 (x - r_1) (x - r_2) \dots (x - r_n)$$

En donde la variable principal es  $x$  y los **coeficientes del polinomio** son todos los valores  $c_i$  desde  $c_1$  hasta  $c_n$ . Al conjunto  $r_i$  desde  $r_1$  hasta  $r_n$ , se les llama **raíces del polinomio** que corresponden a los valores de  $x$  para los que  $f(x) = 0$ .

El llamado **grado del polinomio** corresponde al valor del mayor exponente utilizado en la expresión para la variable  $x$ . Así, un polinomio de grado 3 para  $x$  podría ser:

$$f(x) = x^3 + 0.25 x^2 - 3.5 x + 0.5$$

Un polinomio de grado  $n$  tendrá  $n$  raíces, que pueden ser valores únicos ó repetidos y en forma de números reales ó complejos. Si todos los coeficientes de la función son reales y existen raíces complejas, éstas se presentarán en pares conjugados.

### 5.2.1 Representación de polinomios

Para Matlab, la forma de representar a un polinomio es mediante el uso de un **vector fila** que contiene los valores de los coeficientes de la función  $f(x)$ ; ordenados de izquierda a derecha desde la mayor potencia de  $x$ , hasta el término independiente.

Por ejemplo:

$$P = -4.7 x^5 + 8.2 x^4 - 9.6 x^3 - 5 x^2 + 4.8 x - 6$$

Se representa como:

$$P = [ -4.7 \ 8.2 \ -9.6 \ -5 \ 4.8 \ -6 ];$$

### 5.2.2 Operaciones con polinomios

Matlab provee todos los operadores y funciones para poder realizar aritmética de polinomios y también ofrece funciones para obtener sus raíces, su valuación y diferenciación e integración numéricas:

#### ❖ Suma y resta de polinomios

Podemos obtener la suma y resta de polinomios aplicando directamente los operadores correspondientes de suma ( + ) y resta ( - ) a los dos vectores que contienen los valores de los coeficientes.

La única restricción es que ambos vectores tienen que ser del mismo tamaño; es decir, tienen que representar a polinomios del mismo grado.

Si los vectores de polinomios a sumarse o restarse son de diferente tamaño, podemos “rellenar” con ceros los coeficientes correspondientes a los términos no existentes en el vector más pequeño y así volverlos del mismo tamaño; y por consiguiente del mismo grado, para poder realizar la operación.

Por ejemplo, la suma de los polinomios:

$$\begin{aligned} f(x) &= 3x^4 - 2x^3 + 5x^2 + 3x - 6 \\ + \quad g(x) &= 2x^3 + x + 4 \end{aligned}$$

En Matlab:

```
>>p1 = [ 3 -2 5 3 -6 ];
>>p2 = [ 0 2 0 1 4 ];
>>pr = p1 + p2;
```

Que resulta ser  $pr = [ 3 \ 0 \ 5 \ 4 \ -2 ]$ , equivale a:  
 $h(x) = 3x^4 + 5x^2 + 4x - 2$ .

Otro ejemplo, la resta de

$$\begin{aligned} F \quad f(x) &= 5x^3 + 4x^2 + 3x - 7 \\ - \quad g(x) &= x^2 - 3 \end{aligned}$$

En Matlab:

```
>>p1 = [ 5 4 3 -7 ];
>>p2 = [ 0 1 0 -3 ];
>>pr = p1 - p2;
```

Resulta en  $pr = [ 5 \ 3 \ 3 \ -4 ]$ , que equivale a:  
 $h(x) = 5x^3 + 3x^2 + 3x - 4$ .

### ❖ Multiplicación y División de polinomios

La multiplicación y división de polinomios son operaciones de un mayor grado de complejidad y que en reiteradas ocasiones son susceptibles a errores por la cantidad de operaciones que se realizan en ellas y la cantidad de términos que se manejan.

Matlab tiene las funciones correspondientes a las operaciones de multiplicación y división de polinomios a través de los comandos:

Para la multiplicación de polinomios:

```
[122] >>conv ( a , b );
```

Este comando no proporciona como resultado el vector de coeficientes del producto de los polinomios representados por los vectores  $a$  y  $b$ . Los vectores  $a$  y  $b$  de los polinomios no tienen que ser necesariamente del mismo tamaño. Si el primer polinomio es grado  $m$  y el segundo polinomio es de grado  $n$ , el polinomio producto será de grado  $m + n$ .

Por ejemplo, la multiplicación de

$$\begin{aligned} f(x) &= 2x^3 + 2x^2 - 5x + 3 \\ * \quad g(x) &= x^2 - 3x + 1 \end{aligned}$$

En Matlab:

```
>>a = [ 2 2 -5 3 ];
>>b = [ 1 -3 1 ];
>>p = conv ( a , b );
```

Proporciona el producto  $p = [ 2 \ -4 \ -9 \ 20 \ -14 \ 3 ]$ , que equivale a:  $h(x) = 2x^5 - 4x^4 - 9x^3 + 20x^2 - 14x + 3$ .

La división de polinomios, aún de mayor complejidad, se resuelve fácilmente con el comando:

```
[123] >>[ c , r ] = deconv ( a , b ) ;
```

Este comando permite dividir el polinomio representado por el vector  $a$  entre el polinomio representado por el vector  $b$ .

Al ejecutarse regresa el par de vectores  $c$  y  $r$ ; el primero,  $c$ , contiene los coeficientes del cociente de la división y el segundo,  $r$ , contiene los coeficientes del residuo. Si la división fuese "exacta" el vector residuo  $r$  contendrá ceros.

Por ejemplo, la división de

$$\frac{f(x)}{g(x)} = \frac{3x^4 - 2x^3 + 5x^2 + 3x - 6}{2x^3 + x + 4}$$

En Matlab:

```
>>p1 = [ 3 -2 5 3 -6 ] ;
>>p2 = [ 2 0 1 4 ] ;
>>[ c , r ] = deconv ( p1 , p2 ) ;
```

Genera los vectores  $c = [ 1.5 \ -1 ]$  que equivale al cociente:  $1.5x - 1.5$ , y  $r = [ 0 \ 0 \ 3.5 \ -2 \ -2 ]$  equivalente al residuo:  $3.5x^2 - 2x - 2$ .

### ❖ Raíces de polinomios

Muchos problemas de ciencias e ingeniería requieren de la obtención de las raíces de ecuaciones de la forma  $y = f(x)$ .

Como ya dijimos anteriormente, si  $f(x)$  es un polinomio de grado  $n$ , entonces  $y = f(x)$  tendrá exactamente  $n$  raíces reales, múltiples, complejas ó una mezcla de ellas.

Vale la pena recordar que gráficamente los valores de las raíces de  $y = f(x)$ , donde  $y = 0$ , corresponden a los valores de  $x$  para los cuales la gráfica del polinomio cruza el eje de las abscisas.

La función Matlab para obtener las raíces de un polinomio es:

```
[124] >>roots ( a ) ;
```

Este comando determina las raíces del polinomio representado por el vector  $a$ , y lo hace devolviendo un vector columna con los valores de dichas raíces.

Por ejemplo, las raíces de

$$f(x) = x^4 - 2.75x^3 + 1.5x^2 + 16x - 12$$

En Matlab:

```
>>p1=[ 1 -2.75 1.5 16 -12 ] ;
>>r = roots ( p1 ) ;
```

O escrito de la forma

```
>>r = roots ( [ 1 -2.75 1.5 16 -12 ] );
```

Regresa:

```
r =
    2.0000 + 2.0000i
    2.0000 - 2.0000i
   -2.0000
    0.7500
```

Que son los valores de las raíces del polinomio especificado.

#### ❖ Obtención del polinomio a partir de sus raíces

Matlab permite realizar la operación inversa a la obtención de las raíces de un polinomio; es decir, dados los valores de un cierto número de raíces, obtener los coeficientes del polinomio característico original que las generaría. Esta opción es muy valiosa en múltiples cálculos de ciencias e ingeniería.

La función para generar los coeficientes de un polinomio a partir de sus raíces es:

```
[125] >>poly ( r );
```

Este comando proporciona un vector fila que contiene los coeficientes del polinomio característico original. Sin embargo, habrá que notar que Matlab siempre **normalizará** los coeficientes de modo que el primero sea siempre unitario.

Es decir, cuando Matlab genera el polinomio característico siempre colocará el coeficiente de uno al término con el grado más alto que afecte a x.

Esto se debe a que el polinomio que se genere exclusivamente a partir de sus raíces es independiente en relación a cualquier multiplicador constante y no es posible que se determinen coeficientes distintos de uno para el término de mayor grado.

Por ejemplo, si originalmente se tiene

$$f(x) = 4x^3 + x^2 - 51x + 36$$

Sus raíces obtenidas con la función *roots* son:

```
r = [ -4.0000 3.0000 0.7500 ]
```

Si ahora, tenemos dichos valores y les aplicamos la función *poly* tenemos que:

```
>>p = poly ( r );
```

ó

```
>>p = poly ( [ -4 3 0.75 ] );
```

El comando regresa  $p = [ 1 \ 0.25 \ -12.75 \ 9 ]$ , que equivale a  $g(x) = x^3 + 0.25x^2 - 12.75x + 9$ .

Se observa que los coeficientes de los términos del polinomio resultante, son una cuarta parte de los originales pues han sido normalizados por Matlab.

**Nota:** *roots* y *poly* presentan errores de redondeo y cálculo en los casos de raíces múltiples ó repetidas.

## ❖ Evaluación de polinomios

Cualquier polinomio  $y = f(x)$  puede ser evaluado para cualquier valor específico de abscisa  $x$ . La función que provee Matlab para este fin es:

```
[126] >> polyval ( p , x ) ;
```

Donde  $p$  es el polinomio a evaluar y  $x$  la abscisa en donde se quiere calcular  $y(x)$ . El comando regresará un valor escalar si  $x$  es también un escalar. Si  $x$  es un vector, se regresa un vector fila que contiene todos los valores calculados para  $y(x_i)$  y que será del mismo tamaño que el vector  $x$ .

Si  $x$  es una matriz, se regresa una matriz del mismo tamaño que  $x$  que contiene todos los valores calculados para cada elemento de la matriz.

Por ejemplo, si se tiene el polinomio

$$f(x) = 4x^3 + x^2 - 51x + 36$$

Y se quiere evaluar  $y(0.5)$ , se tiene que:

En Matlab:

```
>>p = [ 4 1 -51 36 ] ;
```

```
>>s = polyval ( p , 0.5 ) ;
```

ó

```
>>s = polyval ( [ 4 1 -51 36 ] , 0.5 ) ;
```

Dá por resultado  $s = 11.25$ ; es decir que el valor de  $y(0.5) = 11.25$ .

Supongamos ahora que se desea evaluar la función en varios valores de abscisa, digamos: .5, 1.5, 2.4 y 3.8. Lo podemos hacer de las dos formas:

```
>>p = [ 4 1 -51 36 ] ;
```

```
>>x = [ .5 1.5 2.4 3.8 ] ;
```

```
>>s = polyval ( p , x ) ;
```

ó

```
>>s = polyval ( [ 4 1 -51 36 ] , [ .5 1.5 2.4 3.8 ] ) ;
```

Lo que se obtiene por resultado es un vector con los elementos:  $s = [ 11.25 \ -24.75 \ -25.344 \ 76.128 ]$ , que indican que,

$$y(.5) = 11.25$$

$$y(1.5) = -24.75$$

$$y(2.4) = -25.344$$

$$y(3.8) = 76.128$$

Esta función es muy útil para los casos en que se tengan que visualizar los datos pues permite obtener las parejas de valores  $(x_i, y_i)$  para proceder a su graficado.

## ❖ Diferenciación de un polinomio

Matlab nos brinda una función cuyo objetivo es proporcionar un vector con los coeficientes correspondientes a la primera derivada de un polinomio dado. Dicha función es:

```
[127] >>polyder ( x ) ;
```

Donde  $x$  es el vector de los coeficientes que representan al vector original.

Por ejemplo si se quiere derivar a

$$f(x) = 4x^3 + 2x^2 - 5x + 6$$

En Matlab:

```
>>p = [ 4 2 -5 6 ] ;
```

```
>>s = polyder ( p ) ;
```

O también podemos escribir:

```
>>s = polyder ( [ 4 2 -5 6 ] ) ;
```

El comando regresa  $s = [ 12 \ 4 \ -5 ]$ , vector que equivale a los coeficientes de la primera derivada del polinomio original:

$$f'(x) = 12x^2 + 4x - 5$$

### ❖ Polinomios e interpolación

En los problemas de interpolación se supone que se cuenta con una colección de puntos  $(x_i, y_i)$  que están relacionados a través de una función  $y = f(x)$ .

Suponemos además que el problema se trata de poder estimar un valor  $f(p)$ , donde  $p$  no corresponde a ninguno de los valores dados, pero de tal forma que su valor sí se encuentra localizado entre dos de los valores  $x$  del conjunto de puntos de datos original.

La técnica matemática para estimar datos entre dos puntos de datos dados se llama **interpolación**.

En Matlab existen básicamente tres tipos de interpolación: **lineal**, **spline cúbica** e **interpolación cúbica**.

Todos los métodos anteriores requieren que los valores dados de  $x$  estén ordenados en forma ascendente; además, la interpolación cúbica requiere que los valores de  $x$  estén equiespaciados.

También es muy importante que los nuevos valores para los que vamos a calcular su valor interpolado, se localicen dentro del rango dado para los valores de  $x$ .

Matlab nos ofrece la siguiente función para el cálculo de valores interpolados:

```
[128] >>interp1 ( x , y , xi ) ;
```

ó

```
>>interp1 ( x , y , xi , 'método' ) ;
```

Donde  $x$  es un vector columna que contiene los valores de  $x$  de los datos originales. El arreglo  $y$  puede ser un vector columna o bien una matriz de  $n$  columnas, pero debe tener exactamente el mismo número de filas que el vector  $x$ .

El valor  $xi$  puede ser un escalar, un vector o una matriz de valores de  $x$  para los cuáles queremos evaluar los valores de  $y = f(x)$  por interpolación.

El '**método**' se refiere al tipo de interpolación que se quiere realizar; es decir, '**linear**' (interpolación lineal), '**spline**' (spline cúbica) ó '**cubic**' (interpolación cúbica).

En la primera forma del comando se asume que el método de interpolación a efectuar será lineal y en la segunda forma se puede seleccionar el método deseado.

El comando `regresa` un escalar, vector o matriz del mismo tamaño que  $x_i$  y que contiene los valores de  $y = f(x)$  interpolados que corresponden a cada valor en  $x_i$ .

Podemos pensar a la interpolación como una operación de **búsqueda en tablas**, en donde la tabla son las parejas  $(x_i, y_i)$  y el comando `interp1` localiza a los elementos de  $x_i$  en  $x$ ,  $y$ , basado en su posición regresa los valores interpolados, calculados a partir de la pareja de valores del arreglo y correspondientes.

Por ejemplo, si se tienen los datos originales:

Temperatura, °C	Volumen, cm <sup>3</sup>
0	0.00
12	23.67
20	31.89
32	48.32
50	65.74

Se desea interpolar los valores de volumen para las temperaturas de 18°C, 25°C y 44°C, utilizando el método de interpolación *spline* cúbica:

En Matlab:

```
>>t = [ 0 ; 12 ; 20 ; 32 ; 50 ];
>>v = [ 0 ; 23.67 ; 31.89 ; 48.32 ; 65.74 ];
>>p = [ 18 ; 25 ; 44 ];
>>r = interp1 ( t , v , p , 'spline' );
```

Regresa:

```
r =
    29.7699
    38.2027
    62.8743
```

Que es el vector columna  $r$ , de los valores interpolados correspondientes al vector  $p$ :

Temperatura, °C	Volumen, cm <sup>3</sup>
18	29.7699
25	38.2027
44	62.8743

#### ❖ Ajuste de polinomios a datos tabulares

Matemáticamente, se dice que un polinomio de grado  $n$  queda determinado en forma única si se definen  $n+1$  puntos.

Es decir, si se cuenta con  $n+1$  puntos de datos tabulares, es posible ajustar un polinomio único de grado  $n$  y determinar, por consiguiente, sus coeficientes con el fin de obtener el polinomio característico.

Los coeficientes del polinomio ajustado se pueden obtener con la función de Matlab:

```
[129] >>polyfit ( x , y , n );
```

En donde  $x$  y  $y$  son la pareja de vectores, del mismo tamaño, que contienen los valores de abscisas y ordenadas de los puntos tabulares correspondientes.

El tercer parámetro  $n$  es un número escalar que indica el grado del polinomio por ajustar.

Este tercer argumento de *polyfit* es muy importante y por lo general se le deberá de asignar un valor de acuerdo con la longitud de los vectores  $x$  y  $y$  menos uno.

Es decir que el valor de  $n$  deberá ser:

$$n = \text{length} ( x ) - 1$$

Porque el grado del polinomio por ajustar es igual al número de puntos de datos menos uno.

Supongamos por ejemplo que se tienen los dos vectores de datos siguientes:

$$\begin{aligned} x &= [ 2.4 \ 3.4 \ 4.8 \ 5.6 ] \\ y &= [ 2.986 \ 4.567 \ 7.214 \ 3.935 ] \end{aligned}$$

Como se observa tienen cuatro elementos, por lo tanto es posible determinar un polinomio único con el grado  $n = 4 - 1 = 3$ , mediante:

$$\gg r = \text{polyfit} ( x , y , n ) ;$$

Con lo que se obtiene el vector  $r$  que contiene los coeficientes del polinomio ajustado:

$$r = [ -0.8911 \ 9.5747 \ -31.2472 \ 35.1474 ]$$

Que equivale a escribir el polinomio como:

$$f(x) = -0.8911 x^3 + 9.5747 x^2 - 31.2472 x + 35.1474$$

### 5.3 Integración numérica en Matlab

Matemáticamente, la *integral* de una función  $g(x)$  en el intervalo cerrado  $[ a , b ]$  se define como el área bajo la curva de  $g(x)$  entre los puntos  $a$  y  $b$ .

En muchas ocasiones a la evaluación numérica de una integral se le denomina *cuadratura*, nombre que proviene de la solución a un antiguo problema de geometría.

El objetivo, entonces, es poder estimar la función original  $f(x)$  a partir de la otra función  $g(x)$  que es la que integraremos en un cierto rango de límites definidos.

Matlab nos proporciona dos funciones de cuadratura para poder evaluar numéricamente integrales de funciones definidas para un cierto rango:

```
[130] >>quad ( 'función' , a , b , tol ) ;
ó
[131] >>quad8 ( 'función' , a , b , tol ) ;
```

En donde '*función*' es el nombre de la función Matlab que regresa un vector de valores de  $g(x)$  cuando se le proporciona un vector  $x$  de abscisas de entrada.

El nombre de '*función*' puede referirse a otra función predefinida en Matlab, o bien a una función escrita y desarrollada por el usuario.

Los parámetros  $a$  y  $b$  se refieren a los límites de integración inferior y superior, respectivamente.

El parámetro *tol*, que es opcional, se refiere a la **tolerancia** o grado de refinamiento de la aproximación que se desee lograr. El cálculo de la aproximación a la integral continuará hasta que el **error relativo** sea menor que la tolerancia:

$$\left| \frac{\text{estimación anterior} - \text{estimación actual}}{\text{estimación anterior}} \right| < \text{tolerancia}$$

Si no se especifica ningún valor distinto de cero para la tolerancia, se utiliza el valor predeterminado de 0.001.

La función *quad* utiliza la regla de Simpson para la aproximación al valor de la integral definida. La función *quad8* usa la cuadratura de Newton-Cotes recursiva de orden 8 para realizar dicha aproximación al valor de la integral.

Cuando se sospeche que la función a integrar presenta algún tipo de “singularidad”, se recomienda que se use *quad8*. Si se detecta una singularidad se despliega un mensaje de advertencia pero de todas formas se regresa la aproximación al valor de la integral, obteniéndose el resultado deseado.

Por ejemplo, evalúe

$$\int_0^1 4e^{-x} dx$$

Utilizando el método de Simpson, con un valor de tolerancia específico de 0.00001.

En Matlab:

```
>>r = quad ( 'fun1' , 0 , 1 , .00001 ) ;
```

En donde ‘*fun1*’ es un archivo *.m* que se definió previamente a la instrucción *quad*, con el contenido:

```
function y = fun1 ( x )
y = 4 * exp ( -x ) ;
```

El valor aproximado que regresa Matlab para la integral es:  $r = 2.5285$ .

Otro ejemplo, evaluar

$$\int_1^3 \sqrt{1 + \frac{1}{x}} dx$$

Usando el método de Newton-Cotes, con el valor de tolerancia predeterminado de 0.001.

En Matlab:

```
>>r = quad8 ( 'fun2' , 1 , 3 ) ;
```

Con el archivo *fun2.m*:

```
function y = fun2 ( x )
y = sqrt ( 1 + 1 ./ x ) ;
```

Nos arroja el resultado de  $r = 2.4855$  para el valor aproximado de la integral.