

Laboratorio 2 MA-33A 2007-1:

Interpolación y Aproximación de Funciones

Gonzalo Hernández - Gonzalo Rios

UCHile - Departamento de Ingeniería Matemática

1 Manejo de Polinomios (30 min)

En esta sesión aprenderemos el manejo de polinomios: crear polinomios, evaluarlos en un punto, encontrar raíces y operaciones aritméticas entre polinomios. Para esto, nos concentraremos en 2 polinomios:

$$\begin{aligned}p(x) &= 5x^5 + 6x^2 + 7x + 3 \\q(x) &= x^{17} + 3x - 1\end{aligned}$$

Veamos como crear estos polinomios en **Matlab**:

1.1 Creación de un polinomio

El tipo de representación que ocupa **Matlab** para los polinomios es el de un vector fila con los coeficientes de potencia mayor a menor, es decir, para crear p y q se ingresan en **Matlab** los siguientes comandos:

```
>> p=[5 0 0 6 7 3]
```

```
p =
```

```
5      0      0      6      7      3
```

```
>> q=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 -1]
```

```
q =
```

```
Columns 1 through 14
```

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

```
Columns 15 through 18
```

```
0      0      3     -1
```

1.2 Evaluación de un Polinomio (10 min)

Para evaluar un polinomio p en un punto x , **Matlab** dispone del comando `polyval`, que se usa de la siguiente forma:

$$\text{polyval}(p, x)$$

ACT1: Utilice este comando de **Matlab** con los polinomios p o q , y escríbalo en el informe en los casos que la variable x sea:

- (a) Un número
- (b) Una variable con un valor asignado
- (c) Un vector
- (d) Una matriz

1.3 Raíces de Polinomios

Las raíces de un polinomios son los puntos en donde la evaluación del polinomio es igual a cero. **Matlab** dispone del comando `roots` que entrega un vector con todas las raíces del polinomio p . La forma de usarlo es la siguiente:

$$\text{roots}(p)$$

Por ejemplo:

```
>> roots(p)
```

```
ans =
```

```
0.8477 + 0.9836i  
0.8477 - 0.9836i  
-0.7393  
-0.4780 + 0.5028i  
-0.4780 - 0.5028i
```

1.4 Operaciones Aritméticas de Polinomios

1.4.1 Adición y Sustracción de Polinomios (15 min)

Dos polinomios del mismo grado pueden ser sumados (o restados) como la suma (o resta) de dos vectores. Cuando los polinomios no tienen el mismo grado se debe crear un vector auxiliar, que contenga el polinomio de menor grado en la parte izquierda, y a la derecha se rellene con ceros hasta obtener la misma dimensión del vector representante del polinomio de mayor grado. En nuestro caso, esto se debe hacer de la siguiente forma:

```
> s=[0 0 0 0 0 0 0 0 0 0 0 0 0 p]
```

```
s =
```

```
Columns 1 through 14
```

```
0    0    0    0    0    0    0    0    0    0    0    0    5    0
```

```
Columns 15 through 18
```

```
0    6    7    3
```

```
>> s=s+q
```

```
s =
```

```
Columns 1 through 14
```

```
1    0    0    0    0    0    0    0    0    0    0    0    5    0
```

```
Columns 15 through 18
```

```
0    6    10    2
```

Obs: La dimensión del vector q es 18 y la del vector p es 6, entonces el vector s tiene $18 - 6 = 12$ ceros antes de poner el vector p .

ACT2: Programe una función llamada `sumap.m`, donde reciba dos polinomios, y entregue la suma de estos. Escriba en el informe el código.

1.4.2 Multiplicación de Polinomios (5 min)

Dos polinomios p y q se pueden multiplicar con la función predefinida de **Matlab** `conv`, que significa convolución:

$$\text{conv}(p, q)$$

Para utilizar esta función no es necesario que los polinomios p y q tengan el mismo orden. Veamos un ejemplo:

```
>> m=conv(p,q)
```

```
m =
```

```
Columns 1 through 14
```

```
5    0    0    6    7    3    0    0    0    0    0    0    0    0
```

```
Columns 15 through 23
```

```
0    0    15   -5    0    18    15    2   -3
```

Como podemos notar, la dimensión del vector m es 23, que corresponde a la suma de las dimensiones de p y q : $18 + 5 = 23$

ACT3: Utilizando los polinomios p y q , el comando `conv` y la función `sumap.m`, encuentre las raíces del polinomio

$$f = p * (p + q) - q^2$$

Escriba en el informe el código usado, y la representación de f .

1.4.3 División de Polinomios

Un polinomio q puede ser dividido por un polinomio p con la función predefinida de **Matlab** *deconv*, que entrega un vector d con la división, y otro vector r con el resto. La forma de utilizar este comando es la siguiente:

$$[d, r] = \text{deconv}(q, p)$$

En nuestro ejemplo:

```
>> [d,r]=deconv(q,p)
```

d =

Columns 1 through 8

```
0.2000      0      0 -0.2400 -0.2800 -0.1200  0.2880  0.6720
```

Columns 9 through 13

```
0.6800 -0.0096 -1.1376 -1.9296 -1.3437
```

r =

Columns 1 through 8

```
0      0      0      0      0      0      0 -0.0000
```

Columns 9 through 16

```
0  0.0000  0.0000      0      0  4.8528 19.5696 24.9821
```

Columns 17 through 18

```
18.1946  3.0310
```

Ahora bien, si intentamos dividir el vector p por el vector q :

```
>> [d,r]=deconv(p,q)
```

d =

```
0
```

r =

```
5      0      0      6      7      3
```

Este resultado se debe a que el vector q tiene mayor grado que p .

2 Graficar Funciones (15 min)

En esta sección aprenderemos a graficar funciones en **Matlab** con algunas especificaciones: color y tipo de línea y marcas de los puntos.

2.1 Especificación del gráfico (15 min)

Al comando plot se le puede agregar otros input:

$$\text{plot}(x,y,'especificacion\ de\ linea','propiedad','valor')$$

2.1.1 Especificación de línea

- 1) Se puede elegir el tipo de línea:

Tipo de línea	Especificación
Sólida (Default)	- (Línea)
Segmentada	- - (Dos líneas)
Punteada	: (Dos puntos)
Segmento Punto	- . (Línea y punto)

- 2) Se puede elegir el color de la línea:

Color de línea	Especificación
Roja	r
Verde	g
Azul (Default)	b
Calipso	c
Magenta	m
Amarilla	y
Negra	k
Blanca	w

- 3) Se puede elegir el tipo de marcas de los puntos:

Tipo de marcas	Especificación
Ninguna (Default)	(Nada)
Signo más	+
Círculos	o
Asteriscos	*
Puntos	.
Cuadrados	s
Diamantes	d
Estrella de 5 puntas	p
Estrella de 6 puntas	h

Observaciones

- Se elige hasta 1 tipo de línea, 1 color de línea y 1 tipo de marcas, y se ponen las 3 representaciones juntas, sin separaciones, en '*especificacion de linea*'. Ejemplo '*r**'
- Si alguna de las propiedades no se elige, entonces se graficará con la propiedad Default.
- El orden de las propiedades no importa.
- Para evitar ambigüedades como '*- .*' que puede ser línea sólida y marcar los puntos con puntos, o la línea segmento-punto, entonces es mejor siempre especificar las 3 propiedades.

2.1.2 Propiedades y valores

Aparte de las especificaciones ya mencionadas, también existen otras propiedades, que se deben poner aparte de la '*especificacion de linea*'. Se debe indicar la '*propiedad*' y su respectivo '*valor*'. Se pueden poner cuantas uno quiera.

Propiedad	Descripción	Valores posibles
linewidth	Ancho de línea	Un número en unidades de puntos (Default 0.5)
markersize	Tamaño de las marcas	Un número en unidades de puntos
markeredgecolor	Color del borde de las marcas	Un color de la tabla 'Color de línea'
markerfacecolor	Color del relleno de las marcas	Un color de la tabla 'Color de línea'

2.1.3 Título y Etiquetas

Al crear un gráfico, se puede escribir un texto como título o un detalle en las coordenadas, con los comandos siguientes:

Comando	Descripción
title('texto')	Escribe texto arriba del gráfico
xlabel('texto')	Escribe texto en el eje x
ylabel('texto')	Escribe texto en el eje y

ACT4 Grafique el polinomio q en el intervalo $[0,4]$ usando 21 puntos, con línea segmentada de ancho 2, de color azul, marcas del tipo de estrella de 5 puntas, de tamaño 10, con color del borde rojo y color de relleno verde. Ponga algún título en el gráfico. Guarde el gráfico como imagen jpg y copíela al informe, junto a los comandos utilizados para graficarla.

2.2 Gráficos Múltiples

Para poder graficar 2 o más funciones en un mismo gráfico, se utiliza el mismo comando plot agregando nuevos vectores después de las especificaciones del primer gráfico:

`plot(x1,y1,'especificacion de linea1','propiedad1','valor1',x2,y2,'especificacion de linea2','propiedad2','valor2')`

Otra forma de graficar 2 o más funciones de un mismo gráfico es con el siguiente comando:

hold on

Al graficar una función, y luego graficar la otra, el primer gráfico se borra y solo queda el último. Para evitar esto, antes de graficar la segunda función, se debe escribir el comando *hold on*, que sobrepone ambos gráficos. Por ejemplo:

```
>> x=(0:0.01:6);
>> y=x.*sin(x.*x);
>> z=x.*cos(x.*x);
>> plot(x,y)
>> hold on
>> plot(x,z)
```

3 Interpolación (25 min)

En esta sección aprenderemos a interpolar en base a datos obtenidos de alguna función (que se puede conocer o no) utilizando el comando *interp1*. Pero antes, aprenderemos a importar datos a **Matlab**.

3.1 Importar y Exportar Datos (10 min)

Es muy común que se tengan los datos guardados en archivos Excel, por lo que nos concentraremos en ese caso.

3.1.1 Importar

Para importar datos desde algún archivo Excel, este debe estar visualizado en el Current Directory de **Matlab**, y luego utilizar el siguiente comando:

$$\text{nombre_variable} = \text{xlsread}('archivo', 'planilla', 'rango')$$

'archivo': Nombre del archivo Excel que se quiere importar los datos.

'planilla': Nombre de la planilla u hoja específica en donde se quieren importar los datos.

'rango': Region de la planilla de la cual se quieren obtener los datos. Se ocupa la notación de Excel. Ejemplo: 'C2:E5'.

3.1.2 Exportar

Para escribir datos a un archivo Excel, este (en el caso que exista) debe estar visualizado en el Current Directory de **Matlab**, y luego utilizar el siguiente comando:

$$\text{xlswrite}('archivo', \text{nombre_variable}, 'planilla', 'rango')$$

Observaciones

- i) En el caso que el archivo no exista, se crea un archivo con el nombre especificado.
- ii) En el caso que la planilla no exista, se crea una planilla con el nombre especificado.
- iii) En el caso que el rango especificado tenga dimensión menor que el de la variable, entonces se copian en el archivo solo los datos en el rango.

ACT5 Importar los datos del archivo Excel 'torresdelpaine.xls', y guardar la primera columna en una variable *x* en forma de vector fila, y la segunda columna en una variable *y* en forma de vector fila. Escriba en el informe los comandos utilizados.

3.2 Obtener puntos de Interpolación (15 min)

La forma de utilizar el comando *interp1* es la siguiente:

$$y_i = \text{interp1}(x, y, x_i, 'metodo')$$

x: Vector que contiene los puntos conocidos de la abscisa

y: Vector que contiene los puntos conocidos de la ordenada.

xi: Vector que contiene los puntos que se quiere interpolar de la abscisa

yi: Vector que contiene los puntos interpolados de la ordenada.

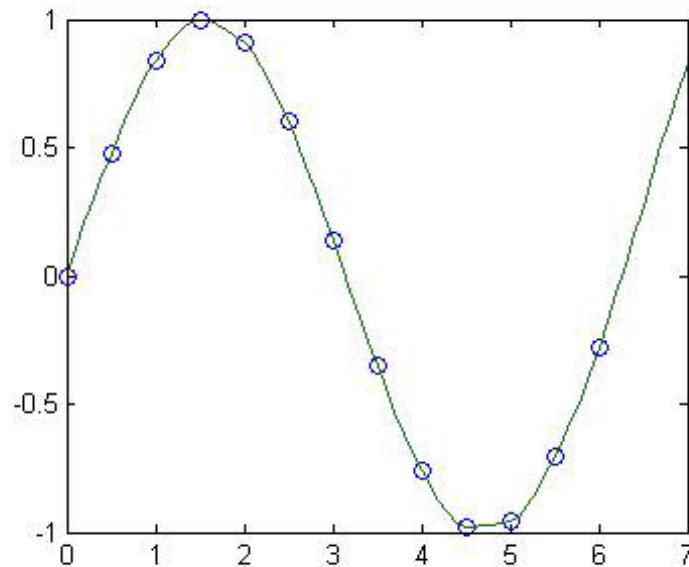
3.2.1 Métodos

'metodo'	Descripción
'linear'	Interpola utilizando polinomios lineales por intervalo (Default)
'nearest'	Entrega el valor conocido más cercano
'spline'	Interpola a través de un Spline Cúbico
'cubic'	Interpola por polinomios cubicos por intervalo

Observaciones

- Los vectores x e y deben tener la misma dimensión.
- Para los métodos '*linear*' y '*nearest*', todos los elementos del vector xi deben estar adentro del intervalo de los valores del vector x . Para los métodos '*spline*' y '*cubic*', se puede extrapolar. Por ejemplo:

```
>> x=(0:0.5:6);
>> y=sin(x);
>> xi=(0:0.1:7);
>> yi=interp1(x,y,xi,'cubic');
>> plot(x,y,'o',xi,yi)
```



ACT6 Utilizando los datos ingresados en las variables x e y , realice una interpolación con el método '*spline*' para luego graficarla, destacando los datos importados. Exporte la interpolación en el mismo archivo Excel, en la planilla '*Spline*'. Escriba en el informe los comandos utilizados y el gráfico. Comente.

4 Aproximación (35 min)

Dados los puntos x_1, \dots, x_m y los valores de alguna función en esos puntos, y_1, \dots, y_m , se desea determinar el polinomio que mejor los representa. Una forma de resolver este problema, es mediante el polinomio de grado n que minimiza el error cuadrático, es decir:

$$\min_{a_0, \dots, a_n} \varepsilon_T = \sum_{i=1}^m [y_i - p_n(x_i)]^2 = \sum_{i=1}^m \left[y_i - \sum_{k=0}^n a_k x_i^k \right]^2$$

4.1 Polyfit (15 min)

Matlab dispone del comando que entrega el polinomio que minimiza el error:

$$p = \text{polyfit}(x, y, n)$$

x: vector con los puntos x_1, \dots, x_m

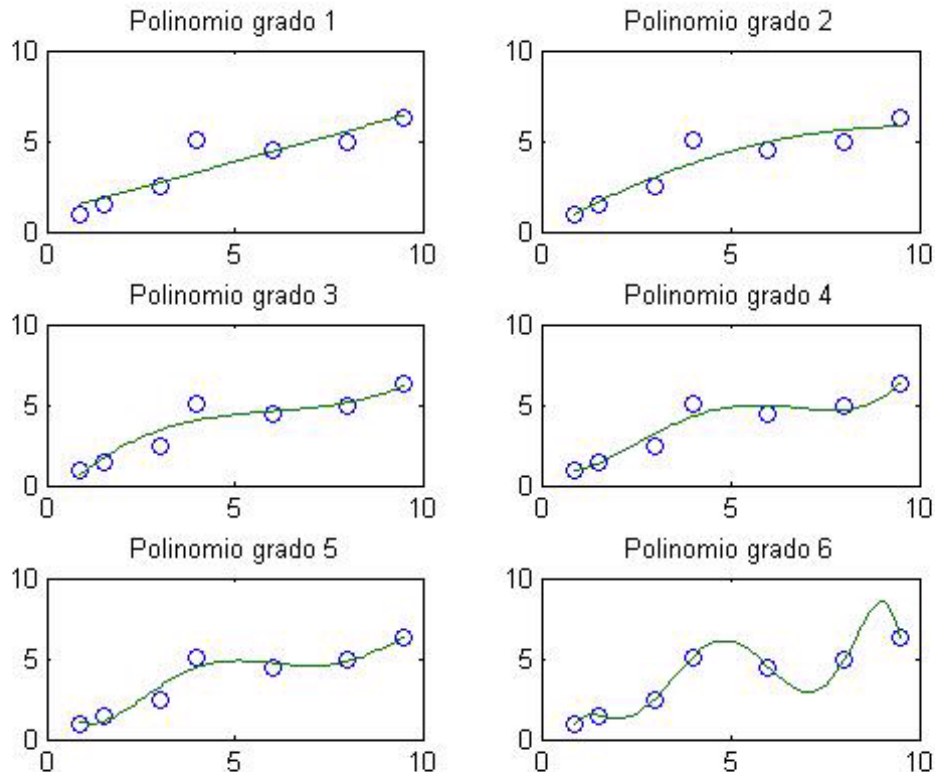
y: vector con los puntos y_1, \dots, y_m

n: grado del polinomio

Ejemplo:

$x = [0.9, 1.5, 3, 4, 6, 8, 9.5];$

$y = [0.9, 1.5, 2.5, 5.1, 4.5, 4.9, 6.3];$



Como podemos observar, no siempre el polinomio pasa por los puntos, pero si el grado es igual a la cantidad de puntos menos uno, entonces el polinomio que minimiza el error cuadrático es el polinomio interpolante, siempre y cuando no hay valores repetidos en el vector x .

ACT7 Programe una función `aproximacion.m` que reciba dos vectores x e y de igual dimensión y devuelva 2 polinomios de aproximación de diferente grado. Escriba en el informe el código. Haga un grafico múltiple con los polinomios entregados. Guarde el gráfico en el informe. Comente

4.2 Errores de Aproximación (20 min)

Dado un polinomio de aproximación de un conjunto de puntos, es muy necesario hacer un análisis de los errores. El error total está dado por:

$$\varepsilon_T = \sum_{i=1}^m [y_i - p_n(x_i)]^2$$

Además, es necesario conocer los errores en cada punto, es decir:

$$\varepsilon_i = [y_i - p_n(x_i)]^2$$

ACT8 Programe una función `errores_aproximacion.m` tal que reciba un vector x , un vector y y un entero n , y que grafique los puntos (x_i, y_i) , el polinomio p de aproximación de grado n , y los errores (x_i, ε_i) . Copie el código en el informe. Ocupando los datos previamente importados en la variables x, y , pruebe esta función con $n = 18$. Guarde en el informe el gráfico y el error total. Comente.

- Bonus: Que la función, además, devuelva el polinomio p de aproximación de grado n y el error total ε_T .