

# Grado Ingeniería Informática.

## Matemáticas 2. Prácticas.

### Practica 6. Interpolación.

Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante

1 de mayo de 2017

Introducción.

Polinomio Interpolador de Lagrange.

Método de Newton.

Splines.

Ejercicios.

# Introducción

## Definición

Se denomina interpolación a la obtención de nuevos puntos partiendo del conocimiento de un conjunto discreto de puntos:

|     |       |       |         |       |
|-----|-------|-------|---------|-------|
| $x$ | $x_0$ | $x_1$ | $\dots$ | $x_n$ |
| $y$ | $y_0$ | $y_1$ | $\dots$ | $y_n$ |

con  $x_i$  todos distintos.

# Introducción

## Problema de Interpolación

Normalmente se calcula una función (polinomio) que pase por los puntos dados.

Un polinomio  $p(x)$  se dice que interpola un conjunto de puntos si  $p(x_i) = y_i$  para  $i = 0, 1, \dots, n$ .

Una variante de este problema es dada una función  $f(x)$ , aproximarla con un polinomio  $p(x)$ . Esto se hace buscando un polinomio interpolador tal que

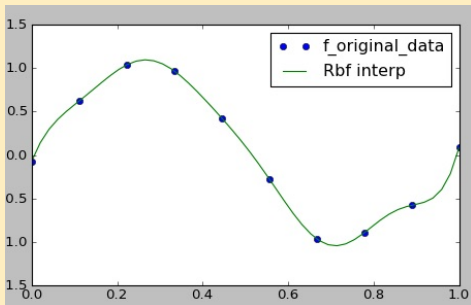
$$\begin{array}{c|c|c|c|c} x & x_0 & x_1 & \dots & x_n \\ \hline f(x) & f(x_0) & f(x_1) & \dots & f(x_n) \end{array}$$

para distintos valores de  $x_i$ .

# Introducción

## Problema de Interpolación

El siguiente gráfico muestra una interpolación con un polinomio de grado 9 para un conjunto de 10 puntos:



# Polinomios de Lagrange

## Definición

Para un conjunto dado de  $n + 1$  puntos  $x_i$ , los  $n + 1$  polinomios de Lagrange  $\ell_i$  están definidos

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}.$$

# Polinomios de Lagrange

## Definición

Se define el polinomio interpolador de Lagrange como

$$p_n(x) = \sum_{i=0}^n y_i \ell_i(x).$$

Si cada polinomio de Lagrange es de grado  $n$ , entonces  $p_n$  también tiene este grado.

Además

$$\ell_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}.$$

# Polinomio Interpolador de Lagrange

## Ejemplo

Obtener el Polinomio Interpolador de Lagrange:

|     |     |       |     |
|-----|-----|-------|-----|
| $x$ | $1$ | $2$   | $3$ |
| $y$ | $3$ | $-10$ | $2$ |

Los polinomios de Lagrange son:

$$\ell_0(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{1}{2}(x-2)(x-3)$$

$$\ell_1(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3)$$

$$\ell_2(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{1}{2}(x-1)(x-2)$$



# Polinomio Interpolador de Lagrange

## Ejemplo (cont)

Finalmente el Polinomio Interpolador de Lagrange es

$$p_2(x) = \frac{3}{2}(x-2)(x-3) + (x-1)(x-3) + (x-1)(x-2)$$

# Polinomio Interpolador de Lagrange

## Ejemplo en Matlab

Comandos *polyfit* y *polyval* para obtener el polinomio interpolador.

```
>> x = [1, 2, 3, 4]
```

```
>> y = [2, 4, 3, 5]
```

```
>> a = polyfit(x, y, 3) % devuelve los coeficientes del  
polinomio interpolador de grado = 3
```

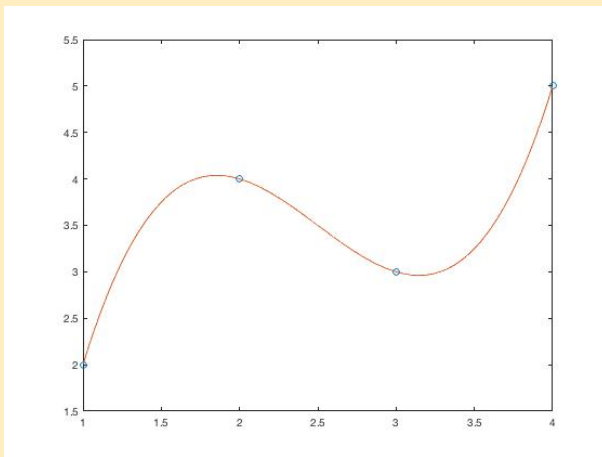
```
>> xp = [1 : 0.01 : 4];
```

```
>> yp = polyval(a, xp); %evalúa el polinomio a en xp
```

```
>> plot(x, y, 'o', xp, yp, '-')
```

# Polinomio Interpolador de Lagrange

## Ejemplo en Matlab



# Método de Newton

## Método de Newton

Supongamos que tenemos datos y el polinomio interpolador de Lagrange, si nos dan un nuevo punto  $(x_{n+1}, y_{n+1})$ , cada polinomio de Lagrange debe ser actualizado lo que conlleva mucho calculo (especialmente si  $n$  es grande).

# Método de Newton

## Método de Newton

La alternativa es construir polinomios de forma iterativa. De esta forma, se crean polinomios  $p_k(x)$  tal que  $p_k(x_i) = y_i$  para  $0 \leq i \leq k$ .

Esto es simple para  $k = 0$ ,

$$p_0(x) = y_0,$$

polinomio constante con valor  $y_0$ .

Suponiendo que se tiene  $p_k(x)$  se quiere obtener  $p_{k+1}(x)$ .

# Método de Newton

## Método de Newton

La siguiente construcción funciona:

$$p_{k+1}(x) = p_k(x) + c(x - x_0)(x - x_1) \cdots (x - x_k),$$

para una constante  $c$ . Nótese que el segundo término será cero para algún  $x_i$  para  $0 \leq i \leq k$ , se tiene que  $p_{k+1}(x)$  interpolará los datos en  $x_0, x_1, \dots, x_k$ . Para obtener el valor de la constante, se calcula  $c$  de forma que

$$y_{k+1} = p_{k+1}(x_{k+1}) = p_k(x_{k+1}) + c(x_{k+1} - x_0) \cdots (x_{k+1} - x_k).$$

# Método de Newton

## Método de Newton

Esta construcción se conoce como Algoritmo de Newton, y el resultado es Método de interpolación de Newton.

# Método de Newton

## Ejemplo

Obtén el polinomio interpolador de los datos

|     |             |   |             |     |             |   |
|-----|-------------|---|-------------|-----|-------------|---|
| $x$ | $\parallel$ | 1 | $\parallel$ | 0,5 | $\parallel$ | 3 |
| $y$ | $\parallel$ | 3 | $\parallel$ | -10 | $\parallel$ | 2 |

usando el Algoritmo de Newton.

$$p_0(x) = 3 \text{ y } p_1(x) = 3 + c(x - 1)$$

Queremos  $-10 = p_1(0,5) = 3 + c(-0,5)$ , y entonces  $c = 26$ .



# Método de Newton

## Ejemplo (cont)

Entonces

$$p_2(x) = 3 + 26(x - 1) + c(x - 1)(x - 0,5)$$

Queremos  $2 = p_2(3) = 3 + 26(2) + c(2)(\frac{5}{2})$ , y entonces  $c = \frac{-53}{5}$ .  
Obtenemos

$$p_2(x) = 3 + 26(x - 1) + \frac{-53}{5}(x - 1)(x - 0,5).$$

# Método de Newton

Es diferente el polinomio de Newton y el de Lagrange?

Los dos métodos dan el mismo polinomio interpolador.

¿Cuál usar? El método de Newton es más flexible y se puede obtener un nuevo polinomio al añadir nuevos datos sin demasiada complicación. Hay una forma de evaluar y almacenar el polinomio de Newton de manera sencilla (en el sentido de número de cálculos requeridos).

# Método de Newton

## Diferencias Divididas

Los coeficientes del polinomio de Newton se pueden calcular de forma sencilla usando *diferencias divididas*.

Asumimos que  $f(x_i)$  at  $x_i$  son conocidos.

# Método de Newton

## Diferencias Divididas

Dada una colección de puntos  $\{(x_i, f(x_i))\}_{i=0}^n$ , las diferencias divididas están definidas de forma recursiva de la siguiente manera:

$$f[x_i] = f(x_i),$$

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

$$f[x_i, \dots, x_{i+k}] = \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k+1}]}{x_{i+k} - x_i}$$

# Método de Newton

## Diferencias Divididas

La forma gráfica es un **tabla piramidal**, donde se computa (de izquierda a derecha):

| $x$   | $f[ ]$   | $f[ , ]$      | $f[ , , ]$         | $f[ , , , ]$            |
|-------|----------|---------------|--------------------|-------------------------|
| $x_0$ | $f[x_0]$ |               |                    |                         |
|       |          | $f[x_0, x_1]$ |                    |                         |
| $x_1$ | $f[x_1]$ |               | $f[x_0, x_1, x_2]$ |                         |
|       |          | $f[x_1, x_2]$ |                    | $f[x_0, x_1, x_2, x_3]$ |
| $x_2$ | $f[x_2]$ |               | $f[x_1, x_2, x_3]$ |                         |
|       |          | $f[x_2, x_3]$ |                    |                         |
| $x_3$ | $f[x_3]$ |               |                    |                         |

La primera columna está formada por  $x_i$  y la segunda por  $f(x_i)$ .

# Método de Newton

## Ejemplo. Diferencias Divididas

Obtener la tabla de diferencias divididas del siguiente conjunto de datos

|        |   |     |   |
|--------|---|-----|---|
| $x$    | 1 | 0,5 | 3 |
| $f(x)$ | 3 | -10 | 2 |

Se empieza escribiendo los datos en la tabla

| $x$ | $f[ ]$ | $f[ , ]$ | $f[ , , ]$ |
|-----|--------|----------|------------|
| 1   | 3      |          |            |
| 0,5 | -10    |          |            |
| 3   | 2      |          |            |

# Método de Newton

## Ejemplo. Diferencias Divididas (cont)

Entonces se calcula:

$$f[x_0, x_1] = \frac{-10 - 3}{0,5 - 1} = 26, \quad y \quad f[x_1, x_2] = \frac{2 - (-10)}{3 - 0,5} = 24/5.$$

Añadiéndolo a la tabla

| $x$ | $f[ ]$ | $f[ , ]$       | $f[ , , ]$ |
|-----|--------|----------------|------------|
| 1   | 3      |                |            |
|     |        | 26             |            |
| 0,5 | -10    |                |            |
|     |        | $\frac{24}{5}$ |            |
| 3   | 2      |                |            |

# Método de Newton

## Ejemplo. Diferencias Divididas (cont)

Entonces, se calcula

$$f[x_0, x_1, x_2] = \frac{24/5 - 26}{3 - 1} = \frac{-53}{5}.$$

Y se completa la tabla:

| $x$ | $f[ ]$ | $f[ , ]$       | $f[ , , ]$      |
|-----|--------|----------------|-----------------|
| 1   | 3      | 26             | $\frac{-53}{5}$ |
| 0,5 | -10    | $\frac{24}{5}$ |                 |
| 3   | 2      |                |                 |

La primera linea contiene los coeficientes del polinomio de la forma de Newton (1, 3, 26 and  $\frac{-53}{5}$ ).



# Implementación de Splines

## Los comandos importantes

Los comandos más relevantes son: **spline**, **ppval**.

Ejemplo:

```
>> qq = spline(x, y);
```

Obtiene el spline cúbico usando los datos  $x, y$  siendo estos las abscisas y las ordenadas respectivamente.

# Implementación de Splines en Matlab

## Ejemplo

```
>> x = 0 : 10;  
>> y = sin(x)  
>> A = spline(x, y);  
A =  
struct with fields :  
form : 'pp'  
breaks : [0 1 2 3 4 5 6 7 8 9 10]  
coefs : [10 × 4 double]  
pieces : 10  
order : 4  
dim : 1
```

# Implementación de Splines en Matlab

## Los coeficientes

Los coeficientes para el spline están en el array *A.coefs*

Vamos a ver cómo utilizar esta estructura. Para ello obtendremos la función cúbica, y la dibujaremos junto con el spline original en el intervalo  $[-1, 10]$ .

```
>> c = A.coefs(3,:); % Coeficientes de la función cubica
```

```
>> xx = linspace(-1,10); % Dominio
```

# Implementación de Splines en Matlab

## Evaluando el polinomio

Podemos evaluar el polinomio manualmente:

$$yy = c(1) * (xx - 2).^3 + c(2) * (xx - 2).^2 + c(3) * (xx - 2) + c(4);$$

Podemos evaluar el polinomio mediante el comando de Matlab `polyval`.

```
>> y2 = polyval(c, xx, [], [2, 1]);
```

Aquí evaluamos el spline:

```
>> y3 = ppval(A, xx);
```

Dibujamos el spline cúbico:

```
>> plot(xx, yy, xx, y3, 'k-');
```

Notese que `y2` y `yy` son iguales:

```
max(abs(y2 - yy))
```

# Interpolación en Matlab

## Comandos

El comando *interp1* se emplea para interpolar una serie de datos. La sintaxis del comando es:

$yi = \text{interp1}(x, y, xi, \text{metodo})$

Donde:

1.  $x$  es la abscisa de los puntos a interpolar, expresada como vector.
2.  $y$  es ordenada de los puntos a interpolar, expresada como vector.
3.  $xi$  son las abscisas para construir la función de interpolación, expresada como vector.
4. método determina el método de interpolación.

# Interpolación en Matlab

## Comandos

El método puede ser uno entre:

- *linear*: interpolación lineal (por defecto).
- *nearest*: interpolación asignando el valor del vecino mas cercano.
- *next*: interpolación asignando el valor del vecino siguiente.
- *previous*: interpolación asignando el valor del vecino anterior.
- *spline*: interpolación con spline cúbico.
- *pchip*: interpolación con polinomios de Hermite.
- *cubic*: igual que *pchip*.
- *v5cubic*: interpolación cúbica usada en Matlab 5.

# Interpolación en Matlab

## Ejemplo

Vamos a hacer la gráfica los diferentes métodos:

```
t = [1 2 3 4 5 6 7 8];
```

```
p = [3 5 7 5 6 7 7 5];
```

```
x = 1 : 0.1 : 8;
```

```
y = interp1(t, p, x, 'spline'); plot(t, p, 'o', x, y); hold on;
```

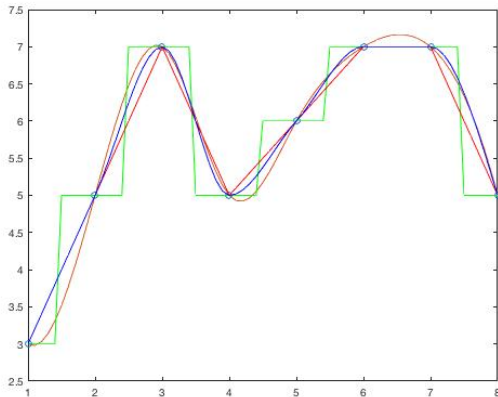
```
y = interp1(t, p, x, 'nearest'); plot(x, y, 'g');
```

```
y = interp1(t, p, x, 'linear'); plot(x, y, 'r');
```

```
y = interp1(t, p, x, 'pchip'); plot(x, y, 'b');
```

# Interpolación en Matlab

## Ejemplo





# Ejercicios

## Práctica #1

Construir una función llamada **lagrange(X,POINTX, POINTY)** para el Método Interpolador de Lagrange. Esta función aproxima la función definida por los puntos  $P1=(POINTX(1),POINTY(1))$ ,  $P2=(POINTX(2),POINTY(2))$ , ...,  $P_N=(POINTX(N), POINTY(N))$  y la calcula en cada elemento de X.

- Si POINTX y POINTY tienen diferente numero de elementos la función debe devolver el valor NaN.
- Probar la función con  $f(x) = x^2$ ,  $x = 0 : 10$ .
- Dibujar  $f(x)$  y los puntos.

# Ejercicios

## Práctica #2

Construye una función llamada **difdiv(x,y)** para el método de Diferencias Divididas. Calcula los componentes de la tabla, donde a partir de la tercera columna se usarán don bucles for.

# Ejercicios

## Práctica #3

**Intenta mejorar la función de la practica anterior para que muestre el polinomio interpolador que se obtiene.**

# Ejercicios

## Práctica #4

**Usa Matlab para obtener el spline cúbico para los datos  $(0, 1), (1, -1), (2, 2), (3, 1), (4, 0)$ .**