⇨ Interpolação Inversa

- Sejam $x_0, x_1, \ldots, x_n$ **nós** pertencentes ao intervalo $[a, b]$ e

  sejam $y_0, y_1, \ldots, y_n$ os **valores nodais** de uma função $f \in C([a, b])$:

x_i	x_0	x_1	$\ldots$	x_n
y_i	y_0	y_1	$\ldots$	y_n

- Se a função f possuir **inversa**, g, podemos escrever

$$y = f(x) \iff x = g(y)$$

- E assim os nós e os valores nodais **trocam de papel**:

y_i	y_0	y_1	$\ldots$	y_n
x_i	x_0	x_1	$\ldots$	x_n

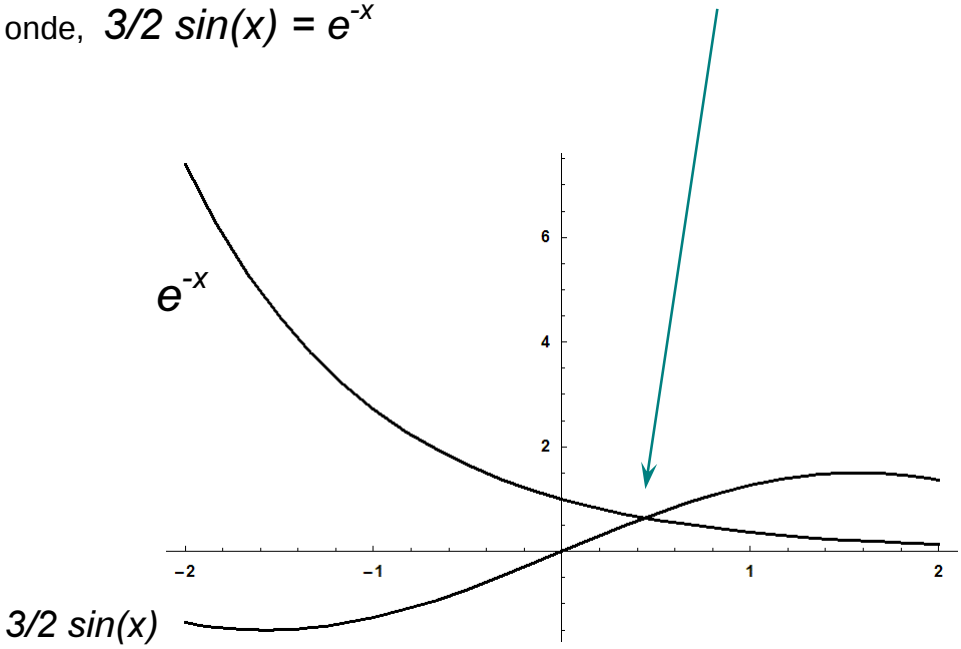
- Podemos pois construir o polinómio interpolador que **interpole os valores nodais** $x_0, x_1, \ldots, x_n$ nos **nós** $y_0, y_1, \ldots, y_n$.

  Neste caso, trata-se de uma **Interpolação Inversa**.

- Utilizamos a Interpolação Inversa, por exemplo, quando **não conhecemos a função inversa** e pretendemos determinar a que valor de x correspondente a um dado $f(x)$.

exemplo: Determinar um valor aproximado do **ponto no intervalo [0,1]**

onde, $3/2\ sin(x) = e^{-x}$



Consideremos a **função diferença**, $f(x) = \frac{3}{2}\sin(x) - e^{-x}$

e procuremos um **zero** de *f(x)* em [0,1] , isto é, um ponto *s* onde *f(s) = 0*

Nos extremos do intervalo: $f(0) = -1$

$f(1) = 0.89433$

a função tem **sinais contrários**, portanto **tem** (pelo menos) uma **raiz**.

E como *f(x)* é **estritamente crescente em [0, 1]** (derivada positiva) então:

- a raiz é **única**
- a função **admite inversa**, *g(y)* , nesse intervalo. Logo,

$$f(s) = 0 \Leftrightarrow s = g(0)$$

Utilizando **interpolação inversa**,

vamos calcular o valor de um **polinómio interpolador de *g(y)*** no ponto ***0***.

Escolhemos alguns **nós no intervalo** e calculamos:

x	0	0.4	0.6	1
$y = f(x)$	−1.00000	−0.08619	0.29815	0.89433

e construindo uma tabela de **diferenças divididas**,

y	x	$x[,]$	$x[,,]$	$x[,,,]$
−1.00000	0			
		0.43773		
−0.08619	0.4		0.06366	
		0.52037		0.04745
0.28815	0.6		0.15356	
		0.67094		
0.89433	1			

obtemos a forma de Newton do **polinómio interpolador**,

$$p(y) = 0 + 0.43773(y+1) + 0.06366(y+1)(y+0.08619) + \\ + 0.04745(y+1)(y+0.08619)(y-0.28815)$$

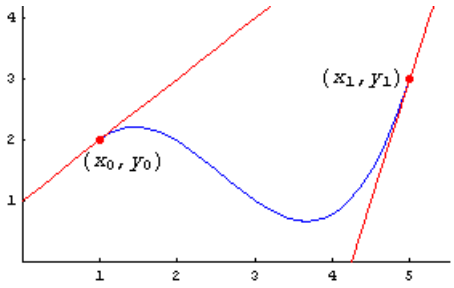
E assim podemos calcular $s = g(0) \approx p(0) = \mathbf{0.44200}$

e verificar que efectivamente, $f(0.44200) = -0.00113 \approx 0$

Portanto ***x = 0.44200*** é uma aproximação do ponto em [0,1] onde $3/2\ \sin(x) = e^{-x}$

⇨ Interpolação de Hermite

Para além dos valores do polinómio interpolador serem **iguais aos valores da função** nos nós de interpolação, vamos exigir também que sejam **iguais os valores das suas derivadas**.



definição: O **Polinómio Interpolador de Hermite** $H_{2n+1}(x)$

satisfaz ***2(n + 1)*** condições:

- Condições sobre a **função** *f* :

$$H_{2n+1}(x_i) = f_i\,, \quad i = 0, 1, \cdots, n$$

- Condições sobre a **derivada** da função *f* :

$$H'_{2n+1}(x_i) = f'_i, \quad i = 0, 1, \cdots, n$$

- Como a função e o polinómio interpolador são **tangentes** nos nós, esta **interpolação** chama-se **osculatória**.
- Demonstra-se que o polinómio H_{2n+1}, de grau menor ou igual a ***2n + 1*** **existe e é único**.

✱ Construção do Polinómio Interpolador de Hermite na forma de Newton

- A construção do polinómio interpolador de Hermite tem por base a **ligação** entre as **diferenças divididas** e as sucessivas **derivadas** da função

- Na **forma de Newton**, substituir os nós $x_0, x_1, \ldots, x_n$

  pelos nós $x_0, x'_0, x_1, x'_1, \ldots, x_n\ x'_n$

  e considerar $x'_k \to x_k$, para $k = 0, 1, \ldots, n$ de tal modo que,

$$\overbrace{\lim_{x'_k \to x_k} f\left[x_k, x'_k\right]}^{f[x_k, x_k]} = \lim_{x'_k \to x_k} \frac{f(x_k) - f(x'_k)}{x_k - x'_k} = f'(x_k)$$

- A interpolação é assim realizada sobre $2n+2$ **pontos** e o **polinómio interpolador de Hermite** de **grau** $2n+1$ na forma de Newton é dado por:

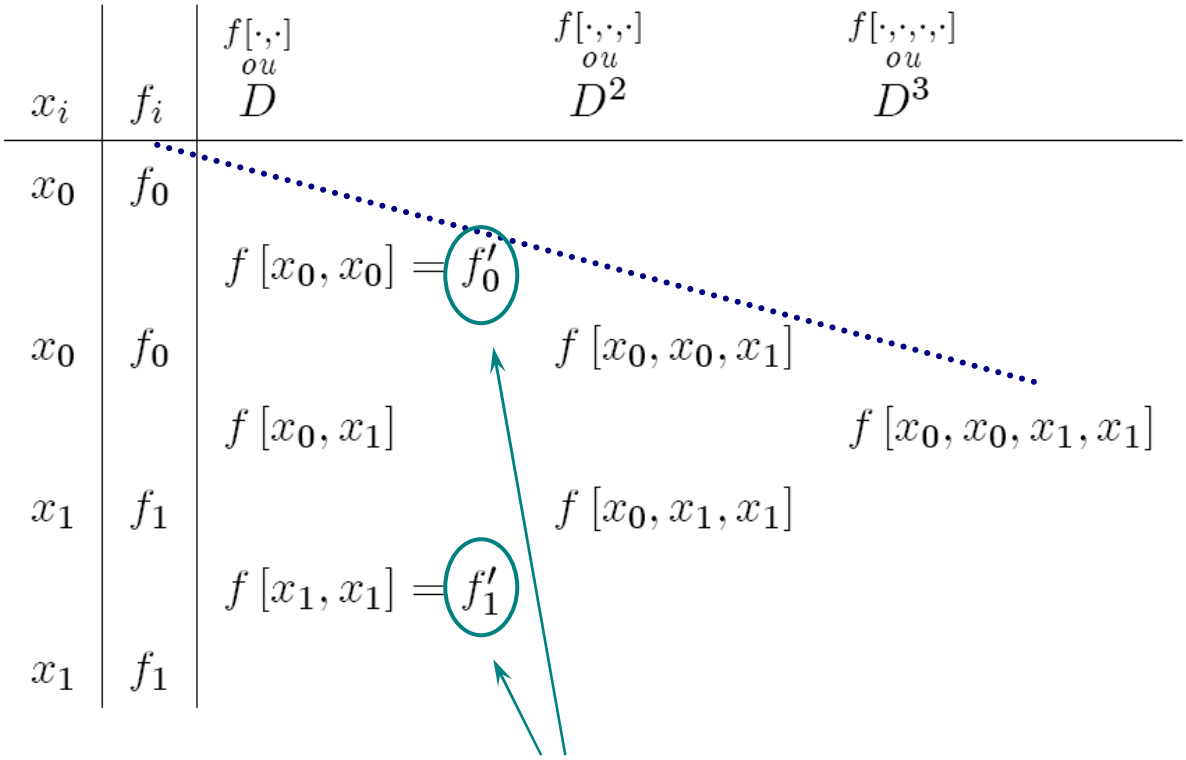
$$H_{2n+1}(x) =$$

$$f(x_0) + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_1](x - x_0)^2 +$$

$$f[x_0, x_0, x_1, x_1](x - x_0)^2 (x - x_1) + \cdots +$$

$$f[x_0, x_0, \cdots, x_n, x_n](x - x_0)^2 (x - x_1)^2 \cdots (x - x_{n-1})^2 (x - x_n)$$

- Por exemplo, para construir o **polinómio cúbico de Hermite**, basta construir a seguinte tabela de **diferenças divididas**, sobre x_0, x_1:



derivadas da função nos pontos

e depois calcular,

$$H_3(x) = f(x_0) + f[x_0,x_0](x-x_0) + f[x_0,x_0,x_1](x-x_0)^2 + f[x_0,x_0,x_1,x_1](x-x_0)^2(x-x_1)$$

exemplo: Com ***f (x) = ln(x)*** , calcular uma aproximação para ***f (1.5)***

usando **interpolação cúbica de Hermite**,

sabendo que ***f (1) = 0*** ***f '(1) = 1***

f (2) = 0.693147 ***f '(2) = 0.5***

Construção da tabela de diferenças divididas:

x_i	f_i	$f[\cdot,\cdot]$ ou D	$f[\cdot,\cdot,\cdot]$ ou D^2	$f[\cdot,\cdot,\cdot,\cdot]$ ou D^3
$x_0 = 1$	$f_0 = 0$			
		$f_0' = 1$		
$x_0 = 1$	$f_0 = 0$		$f[x_0, x_0, x_1] = -0.306853$	
		$f[x_0, x_1] = 0.693147$		$\underbrace{f[x_0, x_0, x_1, x_1]}_{0.113706}$
$x_1 = 2$	$f_1 = 0.693147$		$f[x_0, x_1, x_1] = -0.193147$	
		$f_1' = 0.5$		
$x_1 = 2$	$f_1 = 0.693147$			

Cálculo do **polinómio interpolador cúbico**:

$$\begin{aligned} H_3(x) &= f(x_0) + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_1](x - x_0)^2 + \\ &\quad f[x_0, x_0, x_1, x_1](x - x_0)^2(x - x_1) \\ &= 0 + 1(x-1) - 0.306853(x-1)^2 + 0.113706(x-1)^2(x-2) \end{aligned}$$

donde, $f(1.5) \approx H_3(1.5) =$ ***0.409074***

(e podemos verificar que efectivamente *ln (1.5) = 0.405465...*)

✱ Erro da Interpolação de Hermite

O **teorema geral** do **erro da interpolação polinomial** (pág. 29) continua válido.

Se $f \in C^{2n+2}([a, b])$, para **qualquer** $\widehat{x} \in [a, b]$ **existe** um $\xi \in (a, b)$,

$$e_{2n+1}(\widehat{x}) \equiv f(\widehat{x}) - H_{2n+1}(\widehat{x})$$

$$= \frac{f^{(2n+2)}(\xi)}{(2n+2)!}(\widehat{x} - x_0)^2 (\widehat{x} - x_1)^2 \cdots (\widehat{x} - x_n)^2$$

para o **exemplo** anterior:

$$e_3(1.5) = \frac{f^{(4)}(\xi)}{4!}(1.5 - 1)^2 (1.5 - 2)^2, \ \xi \in (1, 2)$$

ou,

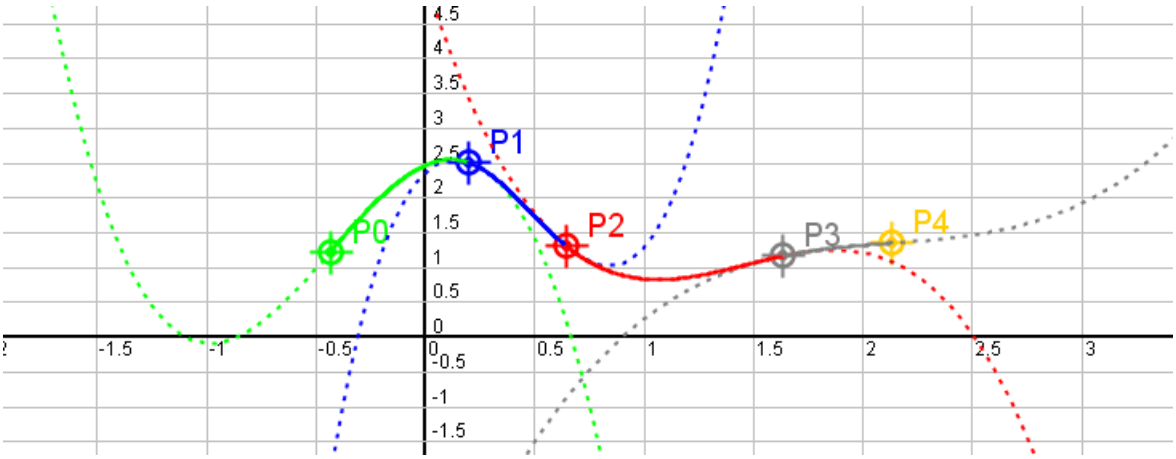
$$|e_3(1.5)| \leq \frac{M}{4!} 0.5^4$$

onde $$M \geq \max_{x \in [1,2]} \left| f^{(4)}(x) \right| = \max_{x \in [1,2]} \left| \frac{-6}{x^4} \right| = 6$$

e portanto $$|e_3(1.5)| \leq 0.2 \times 10^{-1}$$

⇨ Interpolação Polinomial Segmentada

- A interpolação de ***n+1*** nós **com um só polinómio** exige que esse polinómio tenha grau ***n***. Os polinómios de **grau muito elevado** apresentam efeitos indesejáveis.
- Em vez de **um só polinómio** para interpolar **todos os pontos**, a **interpolação segmentada** consiste em utilizar diferentes polinómios interpoladores para **interpolar sucessivos pares** de dados.
- Deste modo, é possível utilizar **polinómios de baixo grau** em cada segmento.



⁕ *Funções Spline*

definição: Uma função S_m é uma **função spline polinomial de grau** m ($m \geq 1$) relativamente aos nós $a = x_0 < x_1 < \ldots < x_n = b$ quando:

- S_m coincide em cada subintervalo $[x_k , x_{k+1}]$ com um polinómio $S_{m,k}$ de **grau menor ou igual** a m , isto é,

$$S_{m,k} = S_m \big|_{[x_k, x_{k+1}]} \in \mathbb{P}_m \qquad k = 0, 1, \ldots, n-1$$

- $S_m \in C^{m-1}[a, b]$

observações:

- **Qualquer polinómio interpolador** de grau ***m*** em **[*a, b*]** é um spline.
- Contudo, tipicamente, um spline é representado através de **diferentes polinómios** de grau **≤ *m*** , um para cada subintervalo
- Um spline pode apresentar **descontinuidades** na sua ***m***-ésima **derivada** nos **nós internos** $x_1, x_2, \ldots, x_{n-1}$.

- As duas condições anteriores **não são suficientes** para **caracterizar univocamente um spline**.

  Vejamos:

  - Podemos representar **cada polinómio** $S_{m,k} = S_m \mid [x_k, x_{k+1}]$ por

$$S_{m,k}(x) = \sum_{i=0}^{m} s_{ik} (x - x_k)^i$$

para $x \in [x_k, x_{k+1}]$, $k = 0, 1, \ldots, n-1$

Logo, temos ***(m + 1) n* coeficientes s_{ik}** para determinar.

  - A partir da **continuidade das derivadas**, temos que

$$S^{(j)}_{m,k-1}(x_k) = S^{(j)}_{m,k}(x_k)$$

para $j = 0, 1, \ldots, m-1$ e $k = 1, 2, \ldots, n-1$

Donde resultam ***m (n − 1)* condições**.

- Como o spline é **interpolador da função** ***f*** nos ***n + 1*** nós,

$$S_m(x_k) = f(x_k) \qquad \text{para} \quad k = 0, 1, 2, \ldots, n$$

temos mais ***n + 1*** **condições**.

- Assim, temos ***(m + 1) n*** **incógnitas**

e apenas ***m (n − 1) + (n + 1) =*** ***(m + 1)n − (m − 1)*** **condições**.

- Restam ***m − 1*** **graus de liberdade.**

- Por isso podemos **impor mais condições** aos splines, como por exemplo, tentar **minimizar a curvatura** fora do intervalo ***[a b]***, como é o caso dos

Splines naturais :

se para $m = 2\,l - 1$, com $l \geq 2$ tivermos,

$$S_m^{(l+j)}(a) = S_m^{(l+j)}(b) = 0, \quad j = 0, 1, \ldots, l-2$$

como veremos para $m = 3$.

✱ *Spline Interpolador Linear*

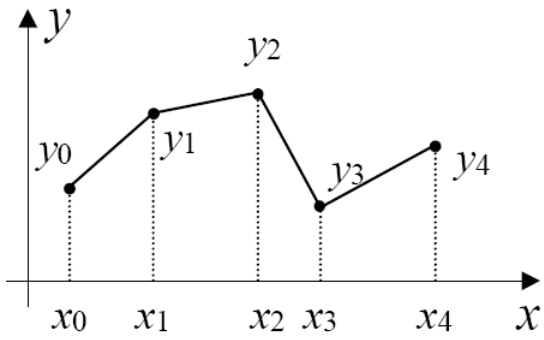
Para o **suporte de interpolação** $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$

com $x_0 < x_1 < \ldots < x_n$

a **função de spline linear**

consiste simplesmente num

caminho poligonal ao longo dos pontos.



- Seja $y_k = f(x_k)$, para $k = 0, 1, \ldots, n$.

  O **segmento de recta** no intervalo $[x_k, x_{k+1}]$ é dado por,

$$S_{1,k}(x) = f(x_k) + f[x_k, x_{k+1}](x - x_k)$$

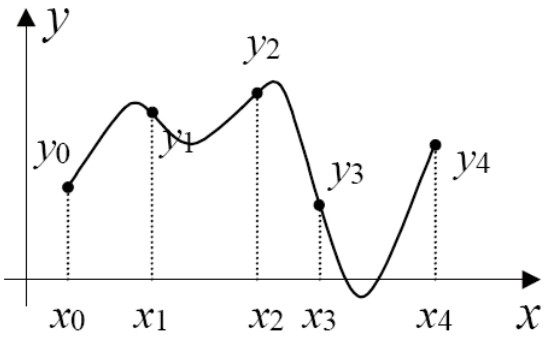
com $$f[x_k, x_{k+1}] = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

- Portanto a **função interpoladora spline linear** pode escrever-se,

$$S_m(x) = \begin{cases} S_{1,0}(x) = f(x_0) + f[x_0, x_1](x - x_0), & x \in [x_0, x_1] \\ S_{1,1}(x) = f(x_1) + f[x_1, x_2](x - x_1), & x \in [x_1, x_2] \\ \vdots & \\ S_{1,n-1}(x) = f(x_{n-1}) + f[x_{n-1}, x_n](x - x_{n-1}), & x \in [x_{n-1}, x_n] \end{cases}$$

- Os “bicos” aparecem porque a função interpoladora **não é diferenciável** em cada nó interno.
- Os **splines lineares** não são obviamente uma boa solução para a maior parte dos problemas reais de interpolação.

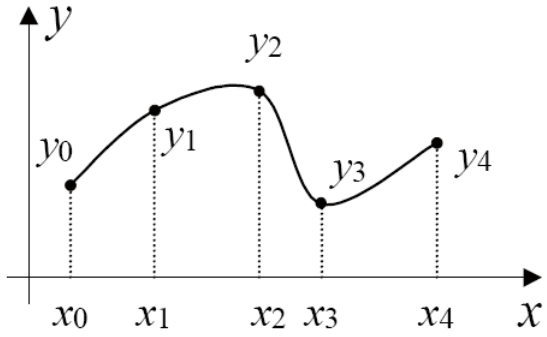
- A **função de spline quadrática** consiste numa sucessão de **arcos de parábola**.



- Os **splines quadráticos** também não são muito utilizados, por geralmente apresentarem **grandes oscilações**.

- Os splines mais utilizados são, sem dúvida, os **splines cúbicos**.

  Cada polinómio tem apenas grau ≤ 3 e tanto a primeira como a segunda derivadas são contínuas em cada nó.



✱ *Spline Interpolador Cúbico*

definição: Consideremos o suporte de interpolação $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$

com $x_0 < x_1 < \ldots < x_n$ e $y_k = f(x_k)$, para $k = 0, 1, \ldots, n$.

Uma função S_3 é um **spline cúbico interpolador** de $f(x)$

nos nós $x_0, x_1, \ldots, x_n$ se existirem n **polinómios** de grau ≤ 3

$S_{3,k}$ com $k = 0, 1, \ldots, n-1$ tais que:

(em cada intervalo, a função spline coincide com um polinómio)

$$S_3(x) = S_{3,k}(x) \qquad \text{para} \qquad x \in [x_k, x_{k+1}],$$
$$k = 0, 1, \ldots, n-1$$

(a função spline interpola nos pontos)

$$S_3(x_k) = f(x_k), \qquad k = 0, 1, \ldots, n$$

(os polinómios ligam-se nos nós internos de forma contínua, sendo também contínuas a primeira e a segunda derivadas)

$$S_{3,k}(x_{k+1}) = S_{3,k+1}(x_{k+1}), \qquad k = 0, 1, \ldots, n-2$$
$$S'_{3,k}(x_{k+1}) = S'_{3,k+1}(x_{k+1}), \qquad k = 0, 1, \ldots, n-2$$
$$S''_{3,k}(x_{k+1}) = S''_{3,k+1}(x_{k+1}), \qquad k = 0, 1, \ldots, n-2$$

- Neste caso ($m = 3$) temos $m - 1 = 2$ **graus de liberdade** o que nos permite escolher **mais 2 condições** a introduzir.

definições: Uma **função spline cúbica** diz-se:

- spline cúbico **natural** se.

$$S''_{3,0}(x_0) = S''_{3,n-1}(x_n) = 0$$

- spline cúbico **completo** se,

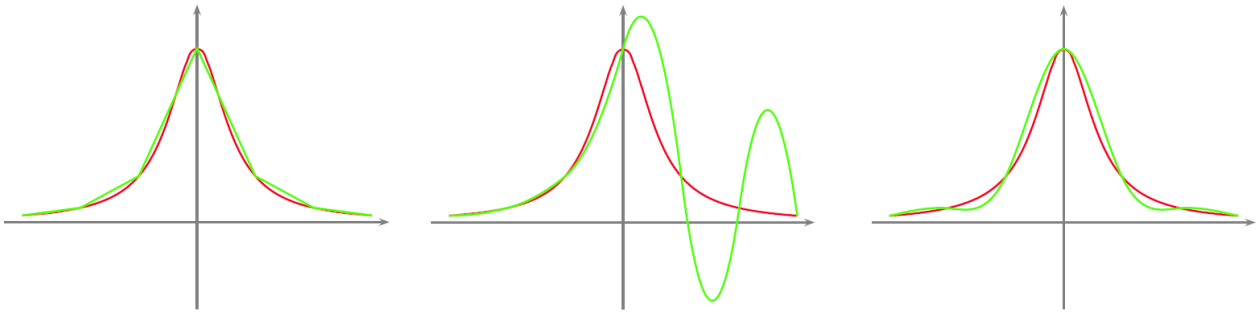
$$S'_{3,0}(x_0) = f'(x_0)$$
$$S'_{3,n-1}(x_n) = f'(x_n)$$

- spline cúbico **periódico** se,

$$S'_{3,0}(x_0) = S'_{3,n-1}(x_n)$$
$$S''_{3,0}(x_0) = S''_{3,n-1}(x_n)$$

exemplos:

da utilização de splines, na interpolação da *função de Runge* com 7 pontos:



linear *quadrático* *cúbico natural*

✱ Erro de Interpolação para Funções Spline Cúbicas

teorema: Sejam $f \in C^{(4)}([x_0, x_n])$ e $x_0 < x_1 < \ldots < x_n$.

e seja S_3 a **função spline cúbica** interpoladora de f em $x_0, x_1, \ldots, x_n$

Se,

$$\max_{x \in [x_0, x_n]} \left| f^{(4)}(x) \right| \leq M$$

e

$$h = \max_{i=0,\ldots,n-1} (x_{i+1} - x_i)$$

então,

$$\max_{x \in [x_0, x_n]} |f(x) - S_3(x)| \leq \frac{5}{384} M h^4$$

e

$$\max_{x \in [x_0, x_n]} \left| f'(x) - S_3'(x) \right| \leq \frac{1}{24} M h^3$$

✱ Construção de um Spline Cúbico

Dado um conjunto de $n+1$ **pontos** $\{ (x_k , f(x_k)) , k = 0,1,\ldots, n \}$

em **cada intervalo** $[x_k , x_{k+1}]$, $k = 0,1,\ldots, n-1$ temos:

$$S_{3,k}(x) = f(x_k) + a_k(x - x_k) + b_k(x - x_k)^2 + c_k(x - x_k)^3$$

$$S_{3,k}(x) = f(x_k) + a_k(x - x_k) + b_k(x - x_k)^2 + c_k(x - x_k)^3$$

onde $a_0, b_0, c_0, a_1, b_1, c_1, \ldots, a_{n-1}, b_{n-1}, c_{n-1}$

são as ***3n* incógnitas** a determinar.

Como se trata de polinómios interpoladores,

$$S_{3,k}(x_k) = f(x_k),\ k = 0, 1, \ldots, n-1$$

e para garantir a **continuidade** nos nós, temos ***n* condições**,

$$S_{3,k}(x_{k+1}) = S_{3,k+1}(x_{k+1})$$

$$S_{3,k}(x_{k+1}) = f(x_{k+1}), \quad k = 0, 1, \ldots, n-1$$

ou, fazendo $h_k = x_{k+1} - x_k$ temos,

$$f(x_k) + a_k h_k + b_k h_k^2 + c_k h_k^3 = f(x_{k+1}), \quad k = 0, 1, \ldots, n-1$$

Para garantir a **continuidade da primeira derivada**, temos mais ***n-1* condições**,

$$S'_{3,k}(x_{k+1}) = S'_{3,k+1}(x_{k+1}) \quad , k = 0, 1, \ldots, n-2$$

Neste caso podemos calcular,

$$S'_{3,k}(x) = a_k + 2b_k(x - x_k) + 3c_k(x - x_k)^2$$

$$S'_{3,k+1}(x) = a_{k+1} + 2b_{k+1}(x - x_{k+1}) + 3c_{k+1}(x - x_{k+1})^2$$

ou seja,

$$a_k + 2b_k h_k + 3c_k h_k^2 = a_{k+1} \quad , k = 0, 1, \ldots, n-2$$

Para garantir a **continuidade da segunda derivada**, temos mais *n-1* **condições**,

calculando, a partir de,

$$S''_{3,k}(x) = 2b_k + 6c_k(x - x_k)$$

$$S''_{3,k+1}(x) = 2b_{k+1} + 6c_{k+1}(x - x_{k+1})$$

temos,

$$\boxed{b_k + 3c_k h_k = b_{k+1}} \quad , k = 0, 1, \ldots, n-2$$

Deste modo, temos *3n* **incógnitas** para *3n-2* **equações**.

A solução mais comum consistem em utilizar **splines cúbicos naturais**, onde,

$$S''_{3,0}(x_0) = S''_{3,n-1}(x_n) = 0$$

ou, $b_0 = 0;$ $\boxed{b_{n-1} + 3c_{n-1}h_{n-1} = 0}$

Deste modo já temos um **sistema linear** de *3n* **equações** a *3n* **incógnitas**.

Por outro lado, podemos calcular,

$$a_k = \frac{f(x_{k+1}) - f(x_k)}{h_k} - b_k h_k - c_k h_k^2, \quad k = 0, 1, \ldots, n-1$$

$$c_k = \frac{b_{k+1} - b_k}{3h_k}, \quad k = 0, 1, \ldots, n-2$$

$$c_{n-1} = -\frac{b_{n-1}}{3h_{n-1}}$$

e substituir em $\boxed{a_k + 2b_k h_k + 3c_k h_k^2 = a_{k+1}}$

Assim, em vez de a_k , b_k, c_k, as **incógnitas** são apenas os b_k, e temos apenas que resolver um **sistema linear** de *n* **equações** a *n* **incógnitas**.

$$\boxed{Hb = g}$$

onde,

$$H = \begin{bmatrix} 1 & & & & & & \\ & 2(h_0+h_1) & h_1 & & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & & \ddots & & & \\ & & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}$$

e $g_0 = 0$

$$g_i = 3\left(\frac{f(x_{i+1})-f(x_i)}{h_i} - \frac{f(x_i)-f(x_{i-1})}{h_{i-1}}\right), \quad i = 1, \ldots, n-1$$

⇨ Interpolação Trigonométrica

- Dada uma função $f : [\, 0,\ 2\pi \,] \rightarrow \mathbb{C}$ **periódica**, isto é, $f(0) = f(2\pi)$

  pretendemos aproximá-la por um **polinómio trigonométrico** φ

  que **interpole** f nos ***n+1*** **nós**, $x_j = \frac{2\pi}{n+1} j,\ j = 0, 1, \ldots n$

$$\varphi(x_j) = f(x_j), \quad j = 0, 1, \ldots, n$$

- Quando ***n*** é **par**, a **função de interpolação trigonométrica** (o polinómio trigonométrico) tem a forma,

$$\varphi(x) = \frac{a_0}{2} + \sum_{k=1}^{M} [a_k \cos(kx) + b_k \sin(kx)]$$

  onde $M = n / 2$

- Quando ***n*** é **ímpar**, a **função de interpolação trigonométrica** tem a forma,

$$\varphi(x) = \frac{a_0}{2} + \sum_{k=1}^{M} [a_k \cos(kx) + b_k \sin(kx)] + a_{M+1} \cos((M+1)x)$$

  onde $M = (n - 1) / 2$

- Recordemos a **fórmula de Euler**,

$$e^{ix} = \cos x + i \sin x$$

- Para ***n* par** podemos reescrever, $\varphi(x) = \frac{a_0}{2} + \sum_{k=1}^{M} [a_k \cos(kx) + b_k \sin(kx)]$ como,

$$\varphi(x) = \sum_{k=-M}^{M} c_k e^{ikx}$$

estando os **coeficientes complexos** a_k, b_k, c_k relacionados por,

$$a_k = c_k + c_{-k}, \quad b_k = i\,(c_k - c_{-k}), \quad k = 0, \ldots, M$$

porque, $\sum_{k=-M}^{k=M} c_k e^{ikx} =$

$$= \sum_{k=-M}^{k=M} c_k \left(\cos(kx) + i \sin(kx)\right)$$

$$= \sum_{k=1}^{k=M} \left[c_k \left(\cos(kx) + i \sin(kx)\right) + c_{-k} \left(\cos(kx) - i \sin(kx)\right)\right] + c_0$$

- De modo análogo para ***n* ímpar** podemos reescrever,

$$\varphi(x) = \sum_{k=-(M+1)}^{M+1} c_k e^{ikx}$$

onde os c_k são os mesmos e $c_{M+1} = c_{-(M+1)} = \frac{a_{M+1}}{2}$

- Juntando os **dois casos**, temos:

$$\varphi(x) = \sum_{k=-(M+\mu)}^{M+\mu} c_k e^{ikx}$$

com $\mu = 0$ se ***n*** par

$\mu = 1$ se ***n*** ímpar

Se a função só tiver **valores reais**, então $c_{-k} = \overline{c_k}$

e nesse caso os **coeficientes** a_k e b_k são todos **reais**.

- Pela sua analogia com as **séries de Fourier**, $\varphi(x)$ chama-se **série discreta de Fourier**.

∗ *Interpolação por séries discretas de Fourier*

- Tomando os pontos $x_j = j\,h$, com $h = 2\pi/(n+1)$ como **nós de interpolação**, temos,

$$\sum_{k=-(M+\mu)}^{M+\mu} c_k e^{ikjh} = f(x_j), \qquad j = 0, \ldots, n$$

- Resta **calcular os valores** dos $c_k,\ k = -(M+\mu), \ldots, M+\mu$

Multipliquemos por $e^{-imx_j} = e^{-imjh}$ onde m é um inteiro entre 0 e n.

e **somemos** para todo o j,

$$\sum_{j=0}^{n} \sum_{k=-(M+\mu)}^{M+\mu} c_k e^{ikjh} e^{-imjh} = \sum_{j=0}^{n} f(x_j)\, e^{-imjh}$$

- Provemos agora um **resultado auxiliar**,

$$\sum_{j=0}^{n} e^{ijh(k-m)} = (n+1)\delta_{km}$$

onde δ_{km} é o delta de Kronecker

- Para $k = m$ ambos os termos são **iguais** a $(n+1)$ e o resultado é verdadeiro.

- Para $k \neq m$ provemos que o somatório se **anula**.

Usando a relação,

$$\begin{aligned} 1 - \left(e^{ih(k-m)}\right)^{n+1} &= \\ &= 1 - e^{ih(k-m)(n+1)} \\ &= 1 - e^{i(k-m)2\pi} \\ &= 1 - \cos\left((k-m)\,2\pi\right) - i\sin\left((k-m)\,2\pi\right) \end{aligned}$$

e somando para todo o j, efectivamente,

$$\sum_{j=0}^{n} e^{ijh(k-m)} = \frac{1 - \left(e^{ih(k-m)}\right)^{n+1}}{1 - e^{ih(k-m)}} = 0$$

quando $k \neq m$

- Retomando o **cálculo dos valores** dos $c_k,\ k = -(M+\mu), \ldots, M+\mu$ na relação,

$$\sum_{j=0}^{n} \sum_{k=-(M+\mu)}^{M+\mu} c_k e^{ikjh} e^{-imjh} = \sum_{j=0}^{n} f(x_j)\, e^{-imjh}$$

e utilizando o resultado auxiliar, temos finalmente,

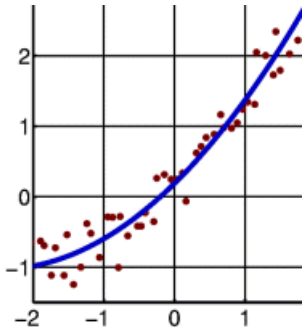
$$c_k = \frac{1}{n+1} \sum_{j=0}^{n} f(x_j)\, e^{-ikjh}$$

$$k = -(M+\mu), \ldots, M+\mu$$

- Para efectuar estes cálculos existe vasta gama de **algoritmos muito eficientes**, de complexidade $O(n \log n)$, conhecidos pelas designação genérica de **FFT (Fast Fourier Transform)**.

- Todos os **sistemas de computação numérica e simbólica**, tais como o MATLAB, possuem funções para calcular **Transformadas de Fourier** e suas **Inversas**, bem como para o cálculo da **interpolação trigonométrica** de um conjunto de pontos e correspondente **visualização gráfica** do resultado.

⇨ Aproximação dos Mínimos Quadrados



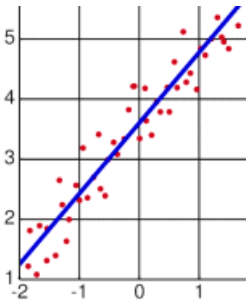
- Em muitos casos reais, os **valores nodais** que temos foram obtidos experimentalmente, vindo portanto **afectados de erros**.

  Em vez de tentar construir uma função interpoladora, faz mais sentido procurar a **função que melhor aproxima** esses valores.

- Seja $\{ (x_i, y_i) \}$, $i = 1, 2, \ldots, m$ um conjunto de pares de números reais onde,

$$y_i \approx f(x_i), \quad i = 1, 2, \ldots, m$$

  A partir deste valores, pretendemos construir uma função que, **de alguma forma**, seja **a melhor aproximação** da função $f(x)$.



- Tomemos **como exemplo o caso linear**, isto é, quando a função interpoladora pretendida for uma recta $y = a\,x + b$.

  Para calcular os parâmetros a e b, podem ser estabelecidos **diferentes critérios**, tais como:

  - Minimizar o **erro máximo**, $\max_{i=1,\ldots,m} |y_i - (ax_i + b)|$
  - Minimizar a **soma dos erros**, $\sum_{i=1}^{m} |y_i - (ax_i + b)|$
  - Minimizar o **erro quadrático**, $\sum_{i=1}^{m} [y_i - (ax_i + b)]^2$

✱ *Funções aproximantes e desvios*

- No caso geral, o problema consiste em determinar a **função que melhor aproxima** um dado conjunto de pontos $\{ (x_i, y_i) \}$, $i = 1, 2, ..., n$.

- A **classe das funções aproximantes** é caracterizada por um conjunto de **parâmetros** $c_1, c_2, \ldots, c_k$.

  Cada função da classe é especificada pelos **valores** desses parâmetros,

$$f(x) = F(x; c_1, c_2, ..., c_k)$$

- Por exemplo,

  - se pretendermos aproximar os pontos por uma **recta**, são dois os parâmetros: c_1 e c_2,

$$f(x) = F(x; c_1, c_2) = c_1 + c_2 x$$

  - se pretendermos aproximar os pontos por uma **parábola**, são três os parâmetros: c_1, c_2 e c_3,

$$f(x) = F(x; c_1, c_2, c_3) = c_1 + c_2 x + c_3 x^2$$

- Para cada classe definem-se os **desvios**, em relação aos valores y_i dos dados,

$$d_i = y_i - F(x_i; c_1, c_2, \ldots, c_k), \quad i = 1, 2, ..., n$$

- Em função dos desvios, é necessário decidir qual o **critério** a estabelecer.

  Cada critério define um **problema de minimização**.

  - Problema de **minimax** (minimização do desvio máximo),

    minimizar $\max_{i=1,\ldots,n} |y_i - F(x_i; c_1, c_2, \ldots, c_k)|$

  - Problema de minimização (da soma) dos **desvios absolutos**,

    minimizar $\sum_{i=1}^{n} |y_i - F(x_i; c_1, c_2, \ldots, c_k)|$

  - Problema de minimização do **erro quadrático total**,

    minimizar $\sum_{i=1}^{n} \left(y_i - F(x_i; c_1, c_2, \ldots, c_k)\right)^2$

    O método de resolução do problema **de minimização do erro quadrático total** chama-se **método dos mínimos quadrados** e a função que o minimiza chama-se **aproximação dos mínimos quadrados**.

⁎ *Método dos Mínimos Quadrados*

- Considere-se uma classe de funções,

$$F(x; c_1, c_2, \ldots, c_k) = c_1 f_1(x) + c_2 f_2(x) \ldots + c_k f_k(x)$$

onde $f_1(x), f_2(x), \ldots, f_k(x)$ são funções dadas.

- A **aproximação dos mínimos quadrados** consiste na determinação dos **parâmetros** $c_1, c_2, \ldots, c_k$ que **minimizam a soma dos quadrados dos desvios**,

$$E(c_1, \ldots, c_k) = \sum_{i=1}^{n} \left(y_i - c_1 f_1(x_i) - c_2 f_2(x_i) - \ldots - c_k f_k(x_i) \right)^2$$

$$= \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{k} c_j f_j(x_i) \right)^2$$

- Tratando-se de um problema de minimização em $\mathbb{R}^k$, para que $E(c_1, c_2, \ldots, c_k)$ seja **mínimo** é necessário que,

$$\nabla E(c_1, \ldots, c_k) = 0 \iff \frac{\partial E}{\partial c_j} = 0, \quad j = 1, \ldots, k$$

- Donde se obtém um **sistema de k equações** a k incógnitas,

$$\begin{cases} c_1 \sum_{i=1}^{n} f_1^2(x_i) + c_2 \sum_{i=1}^{n} f_1(x_i) f_2(x_i) + \ldots + c_k \sum_{i=1}^{n} f_1(x_i) f_k(x_i) = \sum_{i=1}^{n} y_i f_1(x_i) \\ c_1 \sum_{i=1}^{n} f_2(x_i) f_1(x_i) + c_2 \sum_{i=1}^{n} f_2^2(x_i) + \ldots + c_k \sum_{i=1}^{n} f_2(x_i) f_k(x_i) = \sum_{i=1}^{n} y_i f_2(x_i) \\ \\ c_1 \sum_{i=1}^{n} f_k(x_i) f_1(x_i) + c_2 \sum_{i=1}^{n} f_k(x_2) f_2(x_i) + \ldots + c_k \sum_{i=1}^{n} f_k^2(x_i) = \sum_{i=1}^{n} y_i f_k(x_i) \end{cases}$$

- Em certos casos, este sistema tem solução única e permite determinar univocamente os parâmetros $c_1, c_2, \ldots, c_k$ que caracterizam a **melhor função aproximante**.

✱ *Recta dos Mínimos Quadrados (Recta de Regressão)*

- No **caso linear**, **o problema da minimização do erro quadrático** é simples.

  Pretendemos determinar os valores de *a* e de *b* em,

$$F(x; a, b) = a + bx$$

  que minimizam,

$$E(a, b) = \sum_{i=1}^{n} \left(y_i - a - bx_i\right)^2$$

- Para que *E(a, b)* seja **mínimo** é necessário (e prova-se que também suficiente) que,

$$\nabla E(a, b) = 0 \iff \begin{cases} \dfrac{\partial E}{\partial a} = 0 \\ \dfrac{\partial E}{\partial b} = 0 \end{cases}$$

ou seja que,

$$\begin{cases} an & +b\displaystyle\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ a\displaystyle\sum_{i=1}^{n} x_i & +b\displaystyle\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

- Assim temos um sistema linear com **duas equações** (**equações normais**) e as **duas incógnitas** *a* e *b* que caracterizam a recta pretendida (**recta de regressão**).

- Os coeficientes de a e de b e os termos independentes, obtém-se facilmente pela construção de uma tabela,

x_i	y_i	x_i^2	$x_i y_i$
x_1	y_1	x_1^2	$x_1 y_1$
x_2	y_2	x_2^2	$x_2 y_2$
$\dots$	$\dots$	$\dots$	$\dots$
x_n	y_n	x_n^2	$x_n y_n$
$\sum x_i$	$\sum y_i$	$\sum x_i^2$	$\sum x_i y_i$

exemplo: Para determinar a **recta de regressão** que aproxima os pontos,

x	1	2	4	5	7	8	10
y	1	2	4	4	5	6	7

construímos a **tabela**,

	x_i	y_i	x_i^2	$x_i y_i$
	1	1	1	1
	2	2	4	4
	4	4	16	16
	5	4	25	20
	7	5	49	35
	8	6	64	48
	10	7	100	70
$\sum$	37	29	259	194

$$\begin{cases} an & +b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \\ a\sum_{i=1}^{n} x_i & +b\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i \end{cases}$$

donde obtemos o **sistema**,

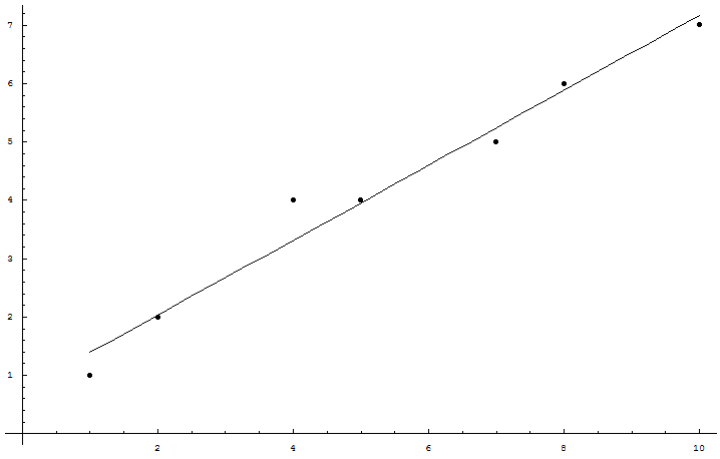
$$\begin{cases} 7\,a + 37\,b = 29 \\ 37\,a + 259\,b = 194 \end{cases}$$

cuja **solução** é $a = 0.75$

$b = 0.\ 6418918918919$

o que permite determinar a **recta de regressão**,

$$y = 0.75 + 0.\ 6418918918919\ x$$



- Em algumas aplicações, são os valores de $\{ x_i \}$, $i = 1, 2, \ldots, n$ que estão **afectados de erros**, sendo os $\{ y_i \}$ **considerados exactos**.

  Nesse caso é necessário efectuar uma **aproximação inversa**.

- Assim, dado $\{ (x_i, y_i) \}$, $i = 1, 2, \ldots, n$ um conjunto de pares de números reais onde,

$$x_i \approx g(y_i), \quad i = 1, 2, \ldots, n$$

  podemos calcular uma **aproximação dos mínimos quadrados** para $g(y)$.

Para o mesmo exemplo:

Basta **trocar os papéis** dos x e y dados,

x	1	2	4	5	7	8	10
y	1	2	4	4	5	6	7

construindo neste caso a **tabela**,

	x_i	y_i	y_i^2	$y_i x_i$
	1	1	1	1
	2	2	4	4
	4	4	16	16
	5	4	16	20
	7	5	25	35
	8	6	36	48
	10	7	49	70
$\sum$	37	29	147	194

donde obtemos o **sistema**,

$$\begin{cases} 7\,a + 29\,b = 37 \\ 29\,a + 147\,b = 194 \end{cases}$$

cuja **solução** é $a = - 0.994680851064$

$b = 1.5159574468085$

o que permite determinar a **recta de regressão inversa**,

$$x = - 0.994680851064 + 1.5159574468085\, y$$

∗ Parábola dos Mínimos Quadrados

- Para aproximar o conjunto de pontos por uma **parábola**, pretendemos determinar os valores de a, b e c em,

$$F(x; a, b, c) = a + bx + cx^2$$

por forma a **minimizar o erro quadrático total**,

$$E(a, b, c) = \sum_{i=1}^{n} \left(y_i - a - bx_i - cx_i^2\right)^2$$

- Para que ocorra o **mínimo** é necessário (e prova-se que também suficiente) que,

$$\nabla E(a, b, c) = 0$$

ou seja,

$$\begin{cases} an + b\sum_{i=1}^{n} x_i + c\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

- Os coeficientes de a de b e de c e os termos independentes, também se obtém pela construção de uma **tabela**,

Para o mesmo exemplo:

x	1	2	4	5	7	8	10
y	1	2	4	4	5	6	7

construímos a **tabela**

	x_i	y_i	x_i^2	x_i^3	x_i^4	$x_i y_i$	$x_i^2 y_i$
	1	1	1	1	1	1	1
	2	2	4	8	16	4	8
	4	4	16	64	256	16	64
	5	4	25	125	625	20	100
	7	5	49	343	2401	35	245
	8	6	64	512	4096	48	384
	10	7	100	1000	10000	70	700
$\sum$	37	29	259	2053	17395	194	1502

donde obtemos o **sistema**,

$$\begin{cases} 7\,a + 37\,b + 259\,c = 29 \\ 37\,a + 259\,b + 2053\,c = 194 \\ 259\,a + 2053\,b + 17395\,c = 1502 \end{cases}$$

cuja **solução** é

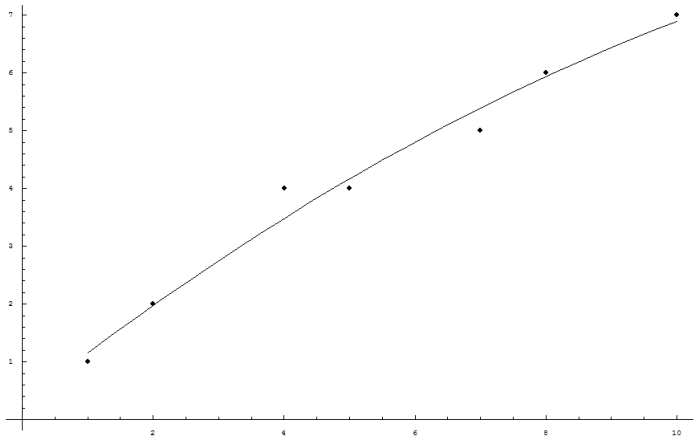
$a = 0.28869047619$

$b = 0.890625$

$c = -0.02306547619$

o que permite determinar a **parábola** que aproxima os pontos,

$$y = 0.28869047619 - 0.890625\, x + 0.02306547619\, x^2$$



∗ *Polinómios de Mínimos Quadrados e outras funções*

- Por vezes é necessário aproximar o conjunto de ***n*** pontos por um **polinómio** geral, de **grau** ***m ≤ n – 1***,

$$p_n(x) = a_0 + a_1 x + \ldots a_m x^m$$

por forma a **minimizar o erro quadrático total**,

$$E(a_0, a_1, \ldots, a_m) = \sum_{i=1}^{n} \left(y_i - p_m(x_i)\right)^2$$

- Tal como nos casos anteriores, para que ocorra o **mínimo** é necessário (e prova-se que também suficiente) que,

$$\nabla E(a_0, a_1, \ldots, a_m) = 0$$

- Desta condição, obtém-se um **sistema linear** de $m + 1$ equações com $m + 1$ incógnitas, também chamadas equações normais.

  Também se prova que este sistema tem uma única solução.

- Contudo, particularmente no caso de polinómios de grau elevado, este sistema tende a ser **mal condicionado**. Por isso são utilizados **polinómios de baixo grau**, ou técnicas mais complexas de reformulação.

- Por vezes é mesmo necessário procurar funções aproximantes que **não são polinómios**, como por exemplo,

$$y = b\ e^{ax} \qquad y = a\ \ln x + b$$

$$y = b\ x^a \qquad y = \frac{a}{x} + b$$

$$y = \frac{a}{x + b}$$

- O problema continua a ser a determinação dos parâmetros a e b.

  Contudo, a **aplicação directa** do Método dos Mínimos Quadrados a estas classes de funções traduz-se pela resolução de **Sistemas de Equações Não Lineares**, cujas soluções são obtidas apenas de forma aproximada.

- Em alternativa, é usual efectuar uma **linearização do problema** (ou **linearização do modelo**).

exemplo: Aproximar por uma função da forma $y = a\, x^b$ os pontos,

x	1	1.2	1.6	2
y	1	1.3	1.4	1.7

Aplicando uma **transformação logarítmica** à fórmula original,

$$\ln y = \ln a + b \ln x$$

e efectuando uma **mudança de variáveis**, $Y = \ln y$

$A = \ln a$

$X = \ln x$

obtemos uma **relação linear**,

$$Y = A + b\, X$$

que aproximamos pelo método do mínimos quadrados.

Construída a tabela, obtemos o sistema,

$$\begin{cases} 4\ A + 1.34547\ \ b = 1.12946 \\ 1.34547\ A + 0.73460\ b = 0.57378 \end{cases}$$

cuja **solução** é, $A = 0.05144 \quad \Rightarrow \quad a = e^A = 1.05247$

$b = 0.68741$

Portanto, a **função aproximante** é,

$$y = 1.05247\, x^{\,0.68741}$$

- Note-se que, os parâmetros assim obtidos **não são óptimos dentro do critério dos mínimos quadrados**. Isto acontece porque eles, de facto, **ajustam o problema linearizado** e não o problema original.

exercício (4):

Considere a seguinte tabela de valores exactos de uma função f,

x	0	1	2
$f(x)$	3	1	4

Sabendo-se que $f'''(x) = 12$ para todo o x, determine o polinómio de grau três interpolador de f.

o problema:

Pretende-se um **polinómio interpolador de grau 3**,

mas apenas são dados **3 nós de interpolação**.

Por outro lado, sabemos que $f'''(x) = 12$ para todo o x.

Como utilizar esta informação?

uma abordagem ingénua:

Se $f'''(x) = 12$ para todo o x, podemos deduzir que,

$$f(x) = 2x^3 + g(x)$$

onde $g(x)$ tem terceira derivada nula e portanto pode ser aproximada por um **polinómio de grau 2**.

De $g(x) = f(x) - 2x^3$ calculamos:

x	0	1	2
$f(x)$	3	1	4
$g(x)$	3	-1	-12

e construímos:

x_i	$g(x_i)$	D^1	D^2
0	3		
		-4	
1	-1		-7/2
		-11	
2	-12		

donde obtemos o polinómio interpolador de grau 2 para $g(x)$,

$$p_2(x) = 3 - 4\,(x-0) - 7/2\,(x-0)\,(x-1)$$
$$= -7/2\,x^2 - 1/2\,x + 3$$

e portanto para $f(x)$ temos o **polinómio de grau 3**,

$$p_3(x) = 2\,x^3 - 7/2\,x^2 - 1/2\,x + 3$$

uma abordagem mais elegante:

Sabemos que (pp. 20-21) existe **pelo menos um ponto** ξ onde,

$$f[x_0, x_1, \cdots, x_n] = \frac{1}{n!} f^{(n)}(\xi)$$

Mas neste caso **sabemos também que** $f'''(x) = 12$ **para todo o** x.

Portanto, **para todo o x**,

$$D^3 = f[x_0, x_1, x_2, x_3] = 12 / 3! = 2$$

e podemos efectivamente construir a tabela:

x_i	$f(x_i)$	D^1	D^2	D^3
0	3			
		-2		
1	1		5/2	
		3		2
2	4		.	
		.		
.	.			

e obter o **polinómio interpolador de grau 3** para $f(x)$,

$$p_3(x) = 3 - 2(x-0) + 5/2(x-0)(x-1) + 2(x-0)(x-1)(x-2)$$
$$= 2x^3 - 7/2 x^2 - 1/2 x + 3$$

exercício (9):

Seja f uma função definida num intervalo $[a, b]$.

(a) Mostre que as diferenças divididas de 1^a ordem $f[x_0, x_1]$ relativas a quaisquer dois pontos distintos $x_0, x_1 \in [a, b]$ são independentes de x_0 e de x_1 se, e só se, f é um polinómio de grau um.

(b) Mostre que se $f(x) = u(x)v(x)$ onde u e v são duas funções definidas em $[a, b]$, então

$$f[x_0, x_1] = u(x_0)v[x_0, x_1] + u[x_0, x_1]v(x_1), \quad x_0, x_1 \in [a, b], \quad x_0 \neq x_1.$$

(c) Supondo que f é um polinómio de 2^o grau com duas raízes reais, deduza das alíneas anteriores o valor das diferenças divididas $f[x_0, x_1]$, $x_0 \neq x_1$.

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

(a) Se $f(x)$ for um **polinómio de grau 1**

uma resposta: $\dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$ é o **declive da recta** que passa pelos dois pontos $(x_0 , f(x_0))$ e $(x_1 , f(x_1))$.

Será **independente dos pontos** se e só se $f(x)$ for uma **recta**.

outra resposta: Sabemos que existe pelo menos um ponto ξ onde,

$$f[x_0, x_1] = f'(\xi)$$

Por isso, $f[x_0, x_1]$ poderá ser **independente dos pontos** x_0 e x_1 considerados apenas quando a **derivada** $f'(x)$ for **constante**, ou seja se e só se $f(x)$ for um **polinómio do 1º grau**.

(b) Se $f(x)=u(x).v(x)$

provemos que $f[x_0, x_1] = u(x_0) \; . \; v[x_0, x_1] \; + \; u[x_0, x_1] \; . \; v(x_1)$

$$u[x_0, x_1] \; = \; \frac{u(x_1) - u(x_0)}{x_1 - x_0}$$

$$v[x_0, x_1] \; = \; \frac{v(x_1) - v(x_0)}{x_1 - x_0}$$

$$f[x_0, x_1] \; = \; \frac{f(x_1) - f(x_0)}{x_1 - x_0} \; = \; \frac{u(x_1).v(x_1) - u(x_0).v(x_0)}{x_1 - x_0}$$

$$= \; \frac{u(x_1).v(x_1) - u(x_0).v(x_0) + u(x_0).v(x_1) - u(x_0).v(x_1)}{x_1 - x_0}$$

$$= \; \frac{-u(x_0).v(x_0) + u(x_0).v(x_1)}{x_1 - x_0} \; + \; \frac{u(x_1).v(x_1) - u(x_0).v(x_1)}{x_1 - x_0}$$

$$= \; u(x_0) \; . \; \frac{v(x_1) - v(x_0)}{x_1 - x_0} \; + \; v(x_1) \; . \; \frac{u(x_1) - u(x_0)}{x_1 - x_0}$$

$$= \; u(x_0) \; v[x_0, x_1] \; + \; u[x_0, x_1] \; v(x_1)$$

{ ou vice versa ...}

(c) Se $f(x)$ for um **polinómio do 2º grau** com **duas raízes reais** α e β, então podemos escrever,

$$f(x) = \underbrace{c\,(x-\alpha)}_{u(x)}\;\underbrace{(x-\beta)}_{v(x)}$$

e considerar, $u(x)$ $v(x)$

de modo a poder utilizar o resultado de **(b)**,

$$f[x_0, x_1] = u(x_0)\,.\,v[x_0, x_1] + u[x_0, x_1]\,.\,v(x_1)$$

Por outro lado $u(x)$ e $v(x)$ são **polinómios de grau 1** portanto, por **(a)**, $u[x_0, x_1]$ e $v[x_0, x_1]$ são **independentes** da escolha de pontos x_0 e x_1.

Então podemos calculá-las nas raízes, isto é, fazer $x_0 = \alpha$ e $x_1 = \beta$.

$$u[x_0, x_1] = \frac{u(x_1) - u(x_0)}{x_1 - x_0}$$

$$v[x_0, x_1] = \frac{v(x_1) - v(x_0)}{x_1 - x_0}$$

Facilmente verificamos que $u[\alpha,\beta] = c$

$$v[\alpha,\beta] = 1$$

E assim, $f[x_0, x_1] = u(x_0)\,.\,v[\alpha,\beta] + u[\alpha,\beta]\,.\,v(x_1)$

$$= c\,(x_0 - \alpha) + c\,(x_1 - \beta)$$

por exemplo: Para $f(x) = 3\,(x - 1)\,(x - 2)$

x_i	$f(x_i)$	D^1	D^2	D^3
0	6			
		-6		
1	0		3	
		0		0
2	0		3	
		6		0
3	6		3	
		12		0
4	18		3	
		21		
6	60			
...				

verifique que, para todo o $k = 0, 1, 2, \ldots$

$$D^1 f(x_k) = f[x_k, x_{k+1}] = 3\,(x_k - 1) + 3\,(x_{k+1} - 2)$$

Note ainda que,

$$D^2 f(x_k) = c \quad \text{para todo o } k = 0, 1, 2, \ldots$$

Porquê?

e que,

$$D^3 f(x_k) = 0 \quad \text{para todo o } k = 0, 1, 2, \ldots$$

Porquê?