Matemática Computacional
Cap. 4 - Ajustamento de dados discretos e Aproximação de funções

Sumário

1 Interpolação polinomial de Lagrange

A interpolação permite encontrar valores entre um par de pontos dados. Consiste em determinar uma função de modo que esta assuma exatamente valores prescritos em certos pontos, os quais se designam por *nós de interpolação*. Em geral, podemos prescrever valores nos nós de interpolação não só para a *função interpoladora*, mas também para as suas derivadas. As funções escolhidas para realizar a interpolação são, em geral, *simples* quanto à sua forma, cálculo, etc. O caso mais comum é aquele em que a função interpoladora é combinação linear de um conjunto de funções linearmente independentes, as quais são escolhidas entre as funções ditas elementares, essencialmente polinómios algébricos e funções trigonométricas.

Além da interpolação polinomial, outros tipos de interpolação podem ser utilizados: interpolação seccional constante, interpolação por splines, interpolação racional, aproximação de Padé, interpolação de Hermite (quando são fornecidos os valores de uma função e das suas derivadas), wavelets, etc. A escolha de um método de interpolação específico depende de critérios como a precisão pretendida para a interpolação, o custo computacional, etc.

A interpolação, e mais geralmente, o ajustamento de dados discretos, é uma ferramenta de *aprendizagem automática* (machine learning). É mais simples do que a maioria dos algoritmos de aprendizagem automática, mas é útil para fazer previsões em muitas situações.

1.1 Representação do polinómio interpolador na base canónica. Sistema com matriz de Vandermonde

Começamos por recordar algumas propriedades básicas dos polinómios algébricos.

Definição 1.1. *Para $n \in \mathbb{N}_0$, designamos por $\mathcal{P}_n$ o espaço linear formado por todos os polinómios*

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

na variável real x e com coeficientes $a_0, ..., a_n$ reais. Um polinómio $p \in \mathcal{P}_n$ diz-se de grau n se $a_n \neq 0$. O conjunto dos monómios $\mathcal{B}_C = \{1, x, ..., x^n\}$ é a base canónica de $\mathcal{P}_n$.

O resultado que se segue é frequentemente usado em questões de unicidade e independência linear relativas a polinómios.

Teorema 1.1. *Se $p \in \mathcal{P}_n$ e tem mais do que n zeros (complexos e contados repetidamente de acordo com a sua multiplicidade) então $p \equiv 0$.*

Sejam $x_0, x_1, ..., x_n \in [a, b]$ nós de interpolação e $y_0, y_1, ..., y_n \in \mathbb{R}$ correspondentes valores nodais, os quais podem ser, por exemplo, os valores de uma função nas abcissas dadas.

A *interpolação polinomial* é o problema que consiste em determinar o polinómio p_n de grau não superior a n tal que

$$p_n(x_k) = y_k, \quad \forall k \in \{0, 1, ..., n\}. \tag{1}$$

Também se usa a designação *interpolação polinomial de Lagrange* para distinguir do caso em que se considera interpolação com condições adicionais para as derivadas do polinómio em alguns nós, caso que é denominado por interpolação de Hermite.

As condições de interpolação (1) para

$$p_n(x) = a_0 + a_1 x + ... + a_n x^n \tag{2}$$

são equivalentes ao sistema linear

$$\begin{bmatrix} 1 & x_0 & ... & x_0^n \\ 1 & x_1 & ... & x_1^n \\ ... & ... & ... & ... \\ 1 & x_n & ... & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ ... \\ y_n \end{bmatrix} \tag{3}$$

Tem-se

Teorema 1.2. *Dados $n+1$ nós de interpolação distintos $x_0, ..., x_n \in [a, b]$ e $n+1$ valores nodais $y_0, ..., y_n \in \mathbb{R}$, existe um e um só polinómio $p_n \in \mathcal{P}_n$ tal que $p_n(x_k) = y_k$, para todo $k \in \{0, 1, ..., n\}$.*

Demonstração. Basta ver que se $y_0 = ... = y_n = 0$ então $a_0 = ... = a_n = 0$. Sejam então $y_0 = ... = y_n = 0$. Isto significa que $p_n(x_k) = 0$, para todo $k \in \{0, 1, ..., n\}$, ou seja p_n tem $n+1$ zeros distintos. Do Teorema 1.1 resulta que $p_n \equiv 0$, ou seja $a_0 = ... = a_n = 0$ na representação (2). □

A matriz do sistema (3) chama-se *matriz de Vandermonde*. É possível verificar diretamente que qualquer matriz de Vandermonde $V(x_0, ..., x_n)$ com nós x_i's distintos entre si é não singular, pois

Lema 1.1. $\det V(x_0, ..., x_n) = \prod_{0 \le i < j \le n} (x_j - x_i)$.

Demonstração. Para $n = 1$, tem-se

$$\det V(x_0, x_1) = \det \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} = x_1 - x_0.$$

Prosseguimos por indução em n. Tem-se

$$\det \begin{bmatrix} 1 & x_0 & ... & x_0^n \\ 1 & x_1 & ... & x_1^n \\ ... & ... & ... & ... \\ 1 & x_n & ... & x_n^n \end{bmatrix} = \det \begin{bmatrix} 1 & x_0 & ... & x_0^n \\ 0 & x_1 - x_0 & ... & x_1^n - x_0^n \\ ... & ... & ... & ... \\ 0 & x_n - x_0 & ... & x_n^n - x_0^n \end{bmatrix}$$

$$= \det \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & x_1 - x_0 & ... & (x_1 - x_0)x_1^{n-1} \\ ... & ... & ... & ... \\ 0 & x_n - x_0 & ... & (x_n - x_0)x_n^{n-1} \end{bmatrix}$$

$$= \prod_{i=1}^{n}(x_i - x_0) \det \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & x_1^{n-1} \\ ... & ... & ... & ... \\ 0 & 1 & ... & x_n^{n-1} \end{bmatrix} = \prod_{i=1}^{n}(x_i - x_0) \det V(x_1, ..., x_n)$$

e neste último determinante usamos a hipótese:

$$\det V(x_1, ..., x_n) = \prod_{1 \le i < j \le n} (x_j - x_i).$$

□

Um *algoritmo* para o cálculo do polinómio interpolador consiste na resolução do sistema linear (3), sendo os valores a_0, ..., a_n encontrados os coeficientes do polinómio interpolador na forma canónica (2).

Exemplo 1.1. *Consideremos os valores apresentados na tabela*

x_k	-1	1	3	4	6
y_k	1	-1	10	2	1

e calculemos os coeficientes a_i do polinómio $p_4(x) = a_0 + a_1x + ... + a_4x^4$ que interpola os valores $y_0, ..., y_4$ nos nós $x_0, ..., x_4$, respetivamente. O sistema linear a resolver é

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 9 & 27 & 81 \\ 1 & 4 & 16 & 64 & 256 \\ 1 & 6 & 36 & 216 & 1296 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 10 \\ 2 \\ 1 \end{bmatrix}.$$

A solução é $(a_0, a_1, a_2, a_3) = (\frac{49}{5}, \frac{57}{20}, \frac{377}{40}, -\frac{77}{20}, \frac{3}{8})$*, pelo que o polinómio interpolador é dado por*

$$p_4(x) = -\frac{49}{5} + \frac{57}{20}x + \frac{377}{40}x^2 - \frac{77}{20}x^3 + \frac{3}{8}x^4.$$

A seguir, iremos deduzir algoritmos alternativos para o cálculo do polinómio interpolador.

O método baseado na matriz de Vandermonde constitui um algoritmo computacionalmente pouco eficiente. A resolução de um sistema com matriz densa tem um custo computacional elevado e o sistema pode apresentar problemas de condicionamento.

1.2 Base de Lagrange. Fórmula interpoladora de Lagrange

Vamos introduzir uma nova base para $\mathcal{P}_n$. Sejam $x_0, ..., x_n$ os nós de interpolação dados (distintos entre si) e, para cada $i \in \{0, ..., n\}$, seja

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

Cada função ℓ_i é um polinómio de grau n, a que se chama *polinómio característico de Lagrange*, e satisfaz

$$\ell_i(x_k) = \delta_{ik} = \begin{cases} 1, & k = i, \\ 0, & k \neq i. \end{cases}$$

Como, para todo $m \in \{0, ..., n\}$, se tem (exercício)

$$x^m = \sum_{i=0}^{n} x_i^m \ell_i(x), \ \forall x \in \mathbb{R},$$

então os $n + 1$ polinómios $\ell_0, ..., \ell_n$ geram $\mathcal{P}_n$, e portanto, formam uma base desse espaço. Ao conjunto $\mathcal{B}_L = \{\ell_0, \ell_1, ..., \ell_n\}$ chama-se *base de Lagrange* de $\mathcal{P}_n$. Mais concretamente, a matriz de Vandermonde é a matriz mudança de base, da base canónica $\mathcal{B}_C$ para a base de Lagrange $\mathcal{B}_L$.

Considerando agora o polinómio interpolador que se pretende determinar representado na base de Lagrange $p_n(x) = b_0\ell_0(x) + ... + b_n\ell_n(x)$ e impondo as condições de interpolação, obtém-se, para cada $k \in \{0, ..., n\}$:

$$p_n(x_k) = y_k \Leftrightarrow \sum_{i=0}^{n} b_i \ell_i(x_k) = y_k \Leftrightarrow \sum_{i=0}^{n} b_i \delta_{ik} = y_k \Leftrightarrow b_k = y_k.$$

Assim, na *representação de Lagrange*, o polinómio interpolador é dado por

$$p_n(x) = \sum_{i=0}^{n} y_i \ell_i(x). \tag{4}$$

Em comparação com o algoritmo baseado na matriz de Vandermonde, a *fórmula interpoladora de Lagrange* (4) envolve cálculos mais diretos, sendo, portanto, um *algoritmo* mais fácil de utilizar.

Exemplo 1.2. *Consideremos novamente os valores apresentados na tabela*

x_k	-1	1	3	4	6
y_k	1	-1	10	2	1

e calculemos agora o polinómio $p_4 \in \mathcal{P}_4$ que interpola os valores $y_0, ..., y_4$ nos nós $x_0, ..., x_4$ através da fórmula interpoladora de Lagrange. Os polinómios de base de Lagrange são

$$\begin{aligned}
\ell_0(x) &= \frac{(x-1)(x-3)(x-4)(x-6)}{(-1-1)(-1-3)(-1-4)(-1-6)} = \frac{1}{280}(x-1)(x-3)(x-4)(x-6),\\
\ell_1(x) &= \frac{(x+1)(x-3)(x-4)(x-6)}{(1+1)(1-3)(1-4)(1-6)} = -\frac{1}{60}(x+1)(x-3)(x-4)(x-6),\\
\ell_2(x) &= \frac{(x+1)(x-1)(x-4)(x-6)}{(3+1)(3-1)(3-4)(3-6)} = \frac{1}{24}(x+1)(x-1)(x-4)(x-6),\\
\ell_3(x) &= \frac{(x+1)(x-1)(x-3)(x-6)}{(4+1)(4-1)(4-3)(4-6)} = -\frac{1}{30}(x+1)(x-1)(x-3)(x-6),\\
\ell_4(x) &= \frac{(x+1)(x-1)(x-3)(x-4)}{(6+1)(6-1)(6-3)(6-4)} = \frac{1}{210}(x+1)(x-1)(x-3)(x-4),
\end{aligned}$$

e o polinómio p_4 é dado (na base de Lagrange) por

$$\begin{aligned}
p_4(x) \;=\; & \frac{1}{280}(x-1)(x-3)(x-4)(x-6) + \frac{1}{60}(x+1)(x-3)(x-4)(x-6) + \\
& + \frac{5}{12}(x+1)(x-1)(x-4)(x-6) - \frac{1}{15}(x+1)(x-1)(x-3)(x-6) + \\
& + \frac{1}{210}(x+1)(x-1)(x-3)(x-4).
\end{aligned}$$

Na forma canónica fica

$$p_4(x) = -\frac{49}{5} + \frac{57}{20}x + \frac{377}{40}x^2 - \frac{77}{20}x^3 + \frac{3}{8}x^4,$$

já obtida com a abordagem anterior, como seria de esperar.

Devido à sua estrutura tão simples, a representação de Lagrange é muito conveniente em questões teóricas relativas à interpolação polinomial e à integração numérica, como veremos no próximo capítulo. No entanto, para cálculos computacionais, esta representação deve ser usada apenas para valores de n pequenos, devido a problemas de estabilidade numérica. Outro inconveniente da representação de Lagrange é o facto de, se um novo nó de interpolação for introduzido, ser necessário recalcular todos os polinómios de base de Lagrange. Neste caso, também a determinação do polinómio interpolador através do sistema com matriz de Vandermonde, terá de ser totalmente refeita.

1.3 Fórmula de Newton com diferenças divididas

A fórmula interpoladora de Lagrange é desaconselhada para valores elevados de pontos a interpolar. Neste caso, é preferível recorrer à fórmula de Newton, que iremos deduzir.

Seja

$$w_0(x) := 1$$

e, para cada $i \in \{1, ..., n\}$, seja

$$w_i(x) := \prod_{j=0}^{i-1}(x - x_j),$$

onde $x_0, ..., x_n$ são os nós de interpolação. Introduzimos a base para $\mathcal{P}_n$

$$\mathcal{B}_N = \{w_0, ..., w_n\},$$

à qual se chama *base de Newton*. Trata-se efetivamente de uma base para $\mathcal{P}_n$, pois escrevendo as componentes dos polinómios w_i em relação à base canónica de $\mathcal{P}_n$ como linhas de uma matriz, obtém-se uma matriz triangular com diagonal unitária, e dado que a dimensão do espaço das linhas dessa matriz é $n+1$, também a expansão linear de $\{w_0, ..., w_n\}$ tem dimensão $n+1$, e portanto, forma uma base de $\mathcal{P}_n$. Note-se ainda que, para cada $i \in \{1, ..., n\}$, se tem $w_i(x_k) = 0$, para todo $k \in \{0, ..., i-1\}$.

Pretende-se então determinar o polinómio interpolador na forma

$$p_n(x) = c_0 + c_1 w_1(x) + ... + c_n w_n(x). \tag{5}$$

Impondo as condições de interpolação (1) em (5):

$$\begin{aligned}
&c_0 = y_0\\
&c_0 + c_1(x_1 - x_0) = y_1\\
&c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2\\
&\dots\\
&c_0 + c_1(x_n - x_0) + ... + c_n(x_n - x_0)...(x_n - x_{n-1}) = y_n
\end{aligned}$$

resulta num sistema linear de matriz triangular inferior cujo determinante é precisamente $\prod_{0\leq i<j\leq n}(x_j - x_i)$.

Note-se que a base de Newton permite construir o polinómio p_n interpolador nos nós $x_0, ..., x_n$, por recorrência, já que

$$p_n(x) = p_{n-1}(x) + c_n w_n(x).$$

onde p_{n-1} é o polinómio interpolador nos nós $x_0, ..., x_{n-1}$. A fórmula de Newton é mais conveniente para valores de n grandes e tem a vantagem de ser recursiva. Se um novo nó de interpolação fôr introduzido, então, para obter o novo polinómio interpolador, apenas será necessário adicionar um termo ao polinómio interpolador nos nós anteriores já calculado.

Relativamente ao cálculos dos coeficientes $c_0, ..., c_n$ tem-se sucessivamente:

$$\begin{aligned}
&c_0 = y_0\\
&c_1 = \frac{y_0}{x_0 - x_1} + \frac{y_1}{x_1 - x_0}\\
&c_2 = ... = \frac{y_0}{(x_0 - x_1)(x_0 - x_2)} + \frac{y_1}{(x_1 - x_0)(x_1 - x_2)} + \frac{y_2}{(x_2 - x_0)(x_2 - x_1)}\\
&\qquad\quad \vdots
\end{aligned}$$

Em geral:

Lema 1.2. *Os coeficientes $c_0, ..., c_n$ são dados por*

$$c_k = \begin{cases} y_0,\ k = 0,\\ \displaystyle\sum_{i=0}^{k} y_i \prod_{j=0, j\neq i}^{k} \frac{1}{x_i - x_j},\ k = 1, ..., n. \end{cases}$$

Demonstração. É claro que c_k é o coeficiente de x^k no polinómio $p_k \in \mathcal{P}_k$ interpolador nos nós $x_0, ..., x_k$. Recorrendo à fórmula interpoladora de Lagrange para escrever p_k,

$$p_k(x) = \sum_{i=0}^{k} y_i \ell_i(x) = \sum_{i=0}^{k} y_i \prod_{j=0, j\neq i}^{k} \frac{x - x_j}{x_i - x_j},$$

é fácil ver que o coeficiente de x^k em p_k é $\displaystyle\sum_{i=0}^{k} y_i \prod_{j=0, j\neq i}^{k} \frac{1}{x_i - x_j}$. □

Pretende-se agora uma expressão recursiva para o cálculo de $c_0, ..., c_n$. Começamos por observar que

$$\begin{aligned} c_0 &= y_0 \\ c_1 &= \frac{y_1 - y_0}{x_1 - x_0} \\ c_2 = ... &= \frac{\dfrac{y_2 - y_1}{x_2 - x_1} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} \\ &\vdots \end{aligned}$$

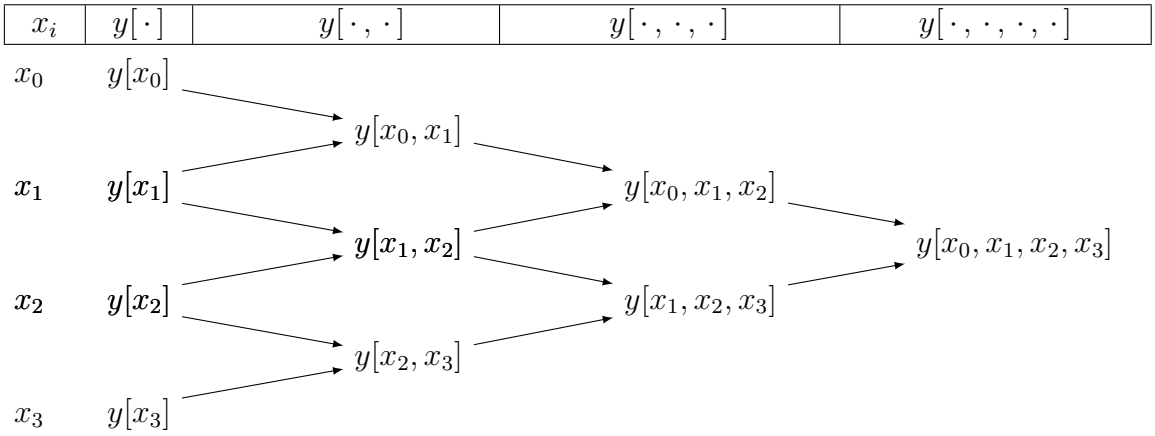
Definição 1.2. *Dados $n+1$ pontos distintos $x_0, ..., x_n \in [a, b]$ e $n+1$ valores $y_0, ..., y_n \in \mathbb{R}$, as* diferenças divididas *$y[x_i, ..., x_{i+k}]$ de ordem k no ponto x_i são definidas recursivamente por*

$$\begin{cases} \text{para } k = 0: \ y[x_i] = y_i, \ i = 0, ..., n \\ \text{para } k = 1, ..., n: \ y[x_i, ..., x_{i+k}] = \dfrac{y[x_{i+1}, ..., x_{i+k}] - y[x_i, ..., x_{i+k-1}]}{x_{i+k} - x_i}, \ i = 0, ..., n-k. \end{cases}$$

Por exemplo,

$$\begin{aligned} y[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}, \quad y[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1}, \ ... \\ y[x_0, x_1, x_2] = \frac{y[x_1, x_2] - y[x_0, x_1]}{x_2 - x_0}, \quad y[x_1, x_2, x_3] = \frac{y[x_2, x_3] - y[x_1, x_2]}{x_3 - x_1}, \ ... \\ y[x_0, x_1, x_2, x_3] = \frac{y[x_1, x_2, x_3] - y[x_0, x_1, x_2]}{x_3 - x_0}, \quad y[x_1, x_2, x_3, x_4] = \frac{y[x_2, x_3, x_4] - y[x_1, x_2, x_3}{x_4 - x_1}, \ ... \\ ... \\ y[x_0, ..., x_n] = \frac{y[x_1, ..., x_n] - y[x_0, ..., x_{n-1}]}{x_n - x_0} \end{aligned}$$

É conveniente organizar o cálculo das diferenças divididas através de uma tabela



que naturalmente se designa por *tabela de diferenças divididas*. Esta é a forma mais eficiente de calcular as diferenças divididas.

Lema 1.3. *As diferenças divididas satisfazem a relação*

$$y[x_i,...,x_{i+k}] = \sum_{m=i}^{i+k} y_m \prod_{j=i,j\neq m}^{i+k} \frac{1}{x_m - x_j}, \quad i = 0,...,n-k, \quad k = 1,...,n.$$

Demonstração. Vamos proceder por indução em k. Para $k = 1$, tem-se

$$y[x_i, x_{i+1}] = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{y_i}{x_i - x_{i+1}} + \frac{y_{i+1}}{x_{i+1} - x_i}.$$

Suponhamos agora que

$$y[x_i,...,x_{i+k-1}] = \sum_{m=i}^{i+k-1} y_m \prod_{j=i,j\neq m}^{i+k-1} \frac{1}{x_m - x_j}, \text{ para } i = 0,...,n-k+1$$

e mostremos que $y[x_i,...,x_{i+k}] = \dfrac{y[x_{i+1},...,x_{i+k}] - y[x_i,...,x_{i+k-1}]}{x_{i+k} - x_i}$ para $i = 0,...,n-k$. Tem-se

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

e

$$\frac{y[x_{i+1},...,x_{i+k}] - y[x_i,...,x_{i+k-1}]}{x_{i+k} - x_i}$$

$$= \frac{\displaystyle\sum_{m=i+1}^{i+k} y_m \prod_{j=i+1, j\neq m}^{i+k} \frac{1}{x_m - x_j} - \sum_{m=i}^{i+k-1} y_m \prod_{j=i, j\neq m}^{i+k-1} \frac{1}{x_m - x_j}}{x_{i+k} - x_i}$$

$$= \frac{\displaystyle\sum_{m=i}^{i+k} y_m (x_m - x_i) \prod_{j=i, j\neq m}^{i+k} \frac{1}{x_m - x_j} - \sum_{m=i}^{i+k} y_m (x_m - x_{i+k}) \prod_{j=i, j\neq m}^{i+k} \frac{1}{x_m - x_j}}{x_{i+k} - x_i}$$

$$= \frac{\displaystyle\sum_{m=i}^{i+k} y_m (x_{i+k} - x_i) \prod_{j=i, j\neq m}^{i+k} \frac{1}{x_m - x_j}}{x_{i+k} - x_i} = \frac{(x_{i+k} - x_i) \displaystyle\sum_{m=i}^{i+k} y_m \prod_{j=i, j\neq m}^{i+k} \frac{1}{x_m - x_j}}{x_{i+k} - x_i}.$$

□

Do Lema 1.3 resulta, para $i = 0$,

$$y[x_0, ..., x_k] = \sum_{m=0}^{k} y_m \prod_{j=0, j\neq m}^{k} \frac{1}{x_m - x_j}$$

e, pelo Lema 1.2, concluimos que $c_k = y[x_0, ..., x_k]$. Portanto, a *representação de Newton* do polinómio interpolador é dada por

$$p_n(x) = y[x_0] + \sum_{i=1}^{n} y[x_0, ..., x_i] \prod_{j=0}^{i-1} (x - x_j).$$

Esta é a chamada *fórmula interpoladora de Newton*. Do Lema 1.3 resulta ainda

Proposição 1.1. *Seja* $E = \{0, 1, ..., n\}$ *e* $\{\sigma(0), ..., \sigma(n)\}$ *uma permutação de* E. *Então* $y[x_{\sigma(0)}, ..., x_{\sigma(n)}] = y[x_0, ..., x_n]$.

Este resultado significa, por exemplo, $y[x_0, x_1] = y[x_1, x_0]$ ou $y[x_0, x_1, x_2] = y[x_1, x_0, x_2] = y[x_2, x_0, x_1]$. O que se pretende realçar é que as diferenças divididas não dependem da ordem em que estão listados os nós que as definem.

Exemplo 1.3. *Retomemos o exemplo anterior, agora para calcular o polinómio interpolador relativo à tabela*

x_k	-1	1	3	4	6
y_k	1	-1	10	2	1

através da fórmula de Newton. A tabela de diferenças divididas fica

$$
\begin{array}{cccccc}
-1 & 1 & & & & \\
 & & \frac{(-1-1)}{1-(-1)} = -1 & & & \\
1 & -1 & & \frac{11/2-(-1)}{3-(-1)} = \frac{13}{8} & & \\
 & & \frac{10-(-1)}{3-1} = \frac{11}{2} & & \frac{-9/2-13/8}{4-(-1)} = -\frac{49}{40} & \\
3 & 10 & & \frac{-8-11/2}{4-1} = -\frac{9}{2} & & \frac{7/5-(-49/40)}{6-(-1)} = \frac{3}{8} \\
 & & \frac{2-10}{4-3} = -8 & & \frac{5/2-(-9/2)}{6-1} = \frac{7}{5} & \\
4 & 2 & & \frac{-1/2-(-8)}{6-3} = \frac{5}{2} & & \\
 & & \frac{1-2}{6-4} = -\frac{1}{2} & & & \\
6 & 1 & & & &
\end{array}
$$

Então

$$
\begin{aligned}
p_4(x) &= y_0 + \sum_{k=1}^{4} y[x_0, ..., x_k] \prod_{i=0}^{k-1} (x - x_i) = \\
&= 1 - (x+1) + \frac{13}{8}(x+1)(x-1) - \frac{49}{40}(x+1)(x-1)(x-3) \\
&\quad + \frac{3}{8}(x+1)(x-1)(x-3)(x-4).
\end{aligned}
$$

2 Aproximação de funções

Aproximar uma função pode ser útil, ou mesmo necessário, em diversas situações. Por exemplo, quando a função é apenas conhecida pelos seus valores em alguns pontos (pode ser somente um número finito de pontos), ou quando a função tem uma forma complicada e se pretende calcular quantidades envolvendo a função, as suas derivadas ou integrais. Assim, é frequente aproximar funções por outras mais simples, em geral, polinómios algébricos.

Começamos por recordar a *aproximação por um polinómio de Taylor*

$$
f(x) \approx \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k
$$

baseada na fórmula de Taylor

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1} \tag{6}$$

válida quando f é de classe C^{n+1} num intervalo que contém x_0 e x. A substituição de f pelo seu polinómio de Taylor pode torna-se computacionalmente dispendiosa por requerer o conhecimento de f e das suas derivadas no ponto x_0. Para além disso, pode acontecer que o polinómio de Taylor não represente com precisão a função f longe de x_0.

2.1 Aproximação de funções através de interpolação polinomial

A principal aplicação da interpolação polinomial consiste na aproximação de funções. Devido à simplicidade das funções polinomiais, cujos cálculos correspondentes apenas envolvem somas e multiplicações, a interpolação polinomial é bastante conveniente, sobretudo do ponto de vista computacional.

Quando os valores y_i a interpolar representam os valores de uma função $f : [a, b] \to \mathbb{R}$ nos pontos x_i, a representação de Lagrange do polinómio interpolador de f é dada por

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

No caso das diferenças divididas, também escrevemos

$$f[x_i] = f(x_i), \quad i = 0, ..., n; \qquad f[x_i, ..., x_{i+k}], \quad i = 0, ..., n-k, \quad k = 1, ..., n,$$

com

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

e a representação de Newton do polinómio interpolador de f é dada por

$$p_n(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, ..., x_i]w_i(x).$$

Segue-se um exemplo para comparação entre vários polinómios interpoladores da função $f(x) := \exp(\sqrt{x^2 - x + 1})\cos(4x)$, incluindo a aproximação por um polinómio de Taylor.

A Fig. 1 representa a função f e três polinómios, interpoladores de f em 5, 8 e 10 nós. Na Fig. 2 mostra-se ainda o polinómio de Taylor de ordem 8 da função f relativo ao ponto $x_0 = 0$.

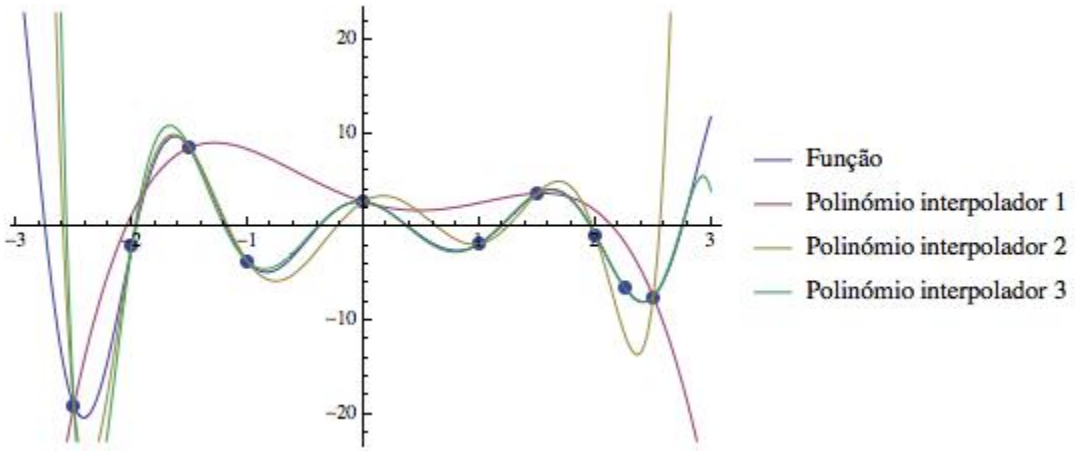


Figura 1: Interpolação polinomial da função $f(x) := \exp(\sqrt{x^2 - x + 1})\cos(4x)$

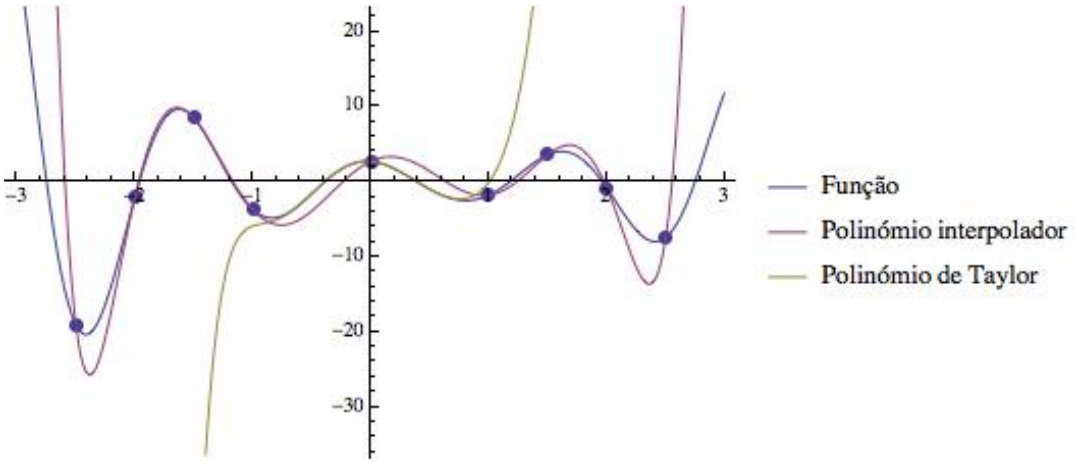


Figura 2: Aproximação polinomial da função $f(x) := \exp(\sqrt{x^2 - x + 1})\cos(4x)$

2.2 Análise do erro na interpolação polinomial

Quando se aproxima uma função, deve ser possível estimar o erro inerente a essa aproximação. Por exemplo, ao aproximar uma função pelo seu polinómio de Taylor (recordar (6)), o erro de interpolação é dado pelo resto de Lagrange:

$$f(x) - \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

Seja $f \in C([a,b])$ e $p_n \in \mathcal{P}_n$ o polinómio interpolador de f nos nós $x_0, ..., x_n \in [a,b]$ (distintos entre si). É claro que a função erro de interpolação $e_n = f - p_n$ satisfaz

$$e_n(x_i) = 0, \, \forall i \in \{1, ..., n\}.$$

Seja agora $x \in [a,b]$ um ponto distinto dos nós de interpolação. Se x for encarado como novo nó, o polinómio interpolador de f em $x_0, ..., x_n, x$ é dado, na forma de Newton, por

$$p_{n+1}(t) = p_n(t) + f[x_0, ..., x_n, x] \prod_{i=0}^{n} (t - x_i).$$

Como $p_{n+1}(x) = f(x)$, tem-se

$$f(x) = p_n(x) + f[x_0, ..., x_n, x] \prod_{i=0}^{n} (x - x_i). \tag{7}$$

Assim,

Teorema 2.1. *Sejam $x_0, ..., x_n \in [a,b]$ $n+1$ nós de interpolação distintos e seja $f \in C([a,b])$. O erro de interpolação pelo polinómio p_n de grau não superior a n é dado por*

$$f(x) - p_n(x) = f[x_0, ..., x_n, x] \prod_{i=0}^{n} (x - x_i).$$

Há uma relação entre as diferenças divididas e as derivadas de f, a qual conhecemos na forma mais simples, fornecida pelo Teorema de Lagrange: dados x_0 e x_1 distintos, existe ξ entre estes dois pontos tal que

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

desde que f seja de classe C^1 no intervalo definido por x_0 e x_1. Em geral, tem-se

Teorema 2.2. *Se $f \in C^n([a,b])$ e $x_0, ..., x_n \in [a,b]$ são distintos entre si, então existe $\xi \in]\min\{x_0, ..., x_n\}, \max\{x_0, ..., x_n\}[$ tal que*

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

Demonstração. Seja $p_n \in \mathcal{P}_n$ o polinómio interpolador de f em $x_0, ..., x_n$ e seja $e_n = f - p_n$. Então a função e_n tem pelo menos $n+1$ zeros em $[\min\{x_0, ..., x_n\}, \max\{x_0, ..., x_n\}]$, e_n' tem pelo menos n zeros em $]\min\{x_0, ..., x_n\}, \max\{x_0, ..., x_n\}[$, e sucessivamente, a derivada $e_n^{(n)}$ tem, pelo menos, um zero:

$$\exists \xi \in]\min\{x_0, ..., x_n\}, \max\{x_0, ..., x_n\}[:\ e_n^{(n)}(\xi) = 0,$$

ou seja,

$$\exists \xi \in]\min\{x_0, ..., x_n\}, \max\{x_0, ..., x_n\}[:\ p_n^{(n)}(\xi) = f^{(n)}(\xi).$$

Agora basta notar que que $p^{(n)}$ é constante e tem o valor $p_n^{(n)} = n! f[x_0, ..., x_n]$. □

Dos Teoremas 2.1 e 2.2 resulta:

Teorema 2.3. *Seja $f \in C^{n+1}[a, b]$. Dados $n+1$ nós distintos $x_0, ..., x_n \in [a, b]$, seja $p_n \in \mathcal{P}_n$ o único polinómio que verifica $p_n(x_j) = f(x_j)$, $j = 0, ..., n$. Então, para cada $x \in [a, b]$, existe*

$$\xi(x) \in]\min\{x_0, ..., x_n, x\}, \max\{x_0, ..., x_n, x\}[$$

tal que

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i). \tag{8}$$

O ponto $\xi(x)$ na expressão do erro de interpolação (8) não é, em geral, conhecido explicitamente. Por isso, na prática, o erro de interpolação polinomial é estimado da seguinte forma

Corolário 2.1. *Nas condições do Teorema 2.3, seja $M_{n+1} := \max_{t \in [a,b]} |f^{(n+1)}(t)|$. Então*

$$|f(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \prod_{i=0}^{n} |x - x_i|.$$

3 Método dos mínimos quadrados

O problema com a interpolação polinomial de grau elevado é a ocorrência de grandes oscilações entre nós de interpolação. Podemos dizer que há demasiado esforço para tentar ajustar exatamente todos os pontos fornecidos e acaba por se perder a estrutura do conjunto de dados. Como consequência, os resultados para as previsões podem ser bastante imprecisos. A este fenómeno chama-se *overfitting*: estamos a tentar ajustar um polinómio de grau maior do que seria necessário para obter um bom ajustamento.

Este problema pode ser resolvido através de um ajustamento segundo o critério dos mínimos quadrados. Na verdade, este método pode ser considerado em duas situações.

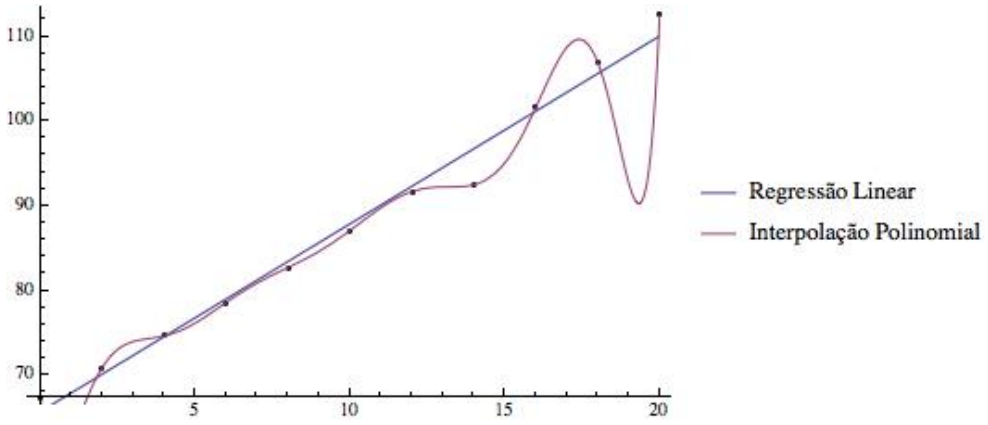


Figura 3: Comportamento de duas funções de ajustamento

1. *Ajustamento de dados discretos*

   Este caso surge quando se pretende ajustar uma função g a um conjunto de pares de números reais

   $$(x_k, y_k),\ k = 1, ..., n.$$

   Cada y_k pode ser um valor (exato ou aproximado) de uma função, i.e. $y_k = y(x_k)$, ou um valor obtido experimentalmente, ou proveniente de cálculos afetados de erros. Neste último caso, a utilização de interpolação para obter g não é razoável. A representação gráfica dos dados pode sugerir o tipo de função de ajustamento apropriado. Segundo o critério dos mínimos quadrados, uma vez decidida a forma da função aproximante g, a qual dependerá de um número finito de parâmetros reais, determina-se os valores desses parâmetros de modo que a soma dos desvios quadrados

   $$D := \sum_{k=1}^{n} [y_k - g(x_k)]^2$$

   seja mínima.

   O ajustamento de uma reta a um conjunto de pontos segundo este critério é designado por *regressão linear*.

   A regressão linear é um algoritmo de *aprendizagem automática* baseado na *aprendizagem supervisionada*. O objectivo da aprendizagem supervisionada é *aprender* a prever uma ou várias variáveis como função de outras variáveis, a partir de um conjunto de exemplos.

2. *Aproximação de uma função num intervalo limitado*

   Neste caso, a função $y = y(x)$ a aproximar está completamente definida num intervalo $[a, b]$ e pretende-se substituir esta função por uma função g que tenha determinadas características, por exemplo, ser facilmente calculável, diferenciável ou primitivável. Então o critério dos mínimos quadrados consiste em determinar g de forma a minimizar

$$D := \int_a^b [y(x) - g(x)]^2 dx.$$

3.1 Sistema normal

Os dois casos descritos podem ser estudados simultaneamente recorrendo a uma formulação comum, após reconhecer

$$D = \|y - g\|^2$$

onde $\|\cdot\|$ representa a norma induzida pelo produto interno

(i)
$$\langle \varphi, \psi \rangle := \sum_{k=1}^{n} \varphi(x_k)\psi(x_k) = \sum_{k=1}^{n} \varphi_k \psi_k$$

no espaço das funções reais definidas no conjunto $\{x_{1,\dots,}x_n\}$, no caso discreto,

(ii)
$$\langle \varphi, \psi \rangle := \int_a^b \varphi(x)\psi(x)dx$$

definido no espaço das funções reais de quadrado integrável em $[a, b]$, no caso contínuo.

Vamos estudar o caso mais simples, em que a função de ajustamento é combinação linear de funções contínuas $\phi_1, ..., \phi_m$ (com $m \leq n$ no caso discreto) selecionadas *a priori*:

$$g = \sum_{i=1}^{m} c_i \phi_i$$

As funções $\phi_1, ..., \phi_m$ são designadas por *funções de base* para o ajustamento. A escolha das funções de base depende dos dados. Por exemplo, no caso de dados periódicos ou quase periódicos, como na observação de marés ou mudanças climáticas, são usados polinómios trigonométricos

$$p(x) = \sum_{i=1}^{n} [a_i \cos(\alpha_i x) + b_i \sin(\beta_i x)]$$

para efetuar o ajustamento.

Vejamos então em que condições é que a função D tem um ponto de mínimo e como o determinar. Tem-se

$$\begin{aligned} D(c_1,...,c_m) &= \|y-g\|^2 = \langle y-g, y-g\rangle = \langle y,y\rangle - 2\langle y,g\rangle + \langle g,g\rangle = \\ &= \langle y,y\rangle - 2\langle y, \sum_{i=1}^m c_i\phi_i\rangle + \langle \sum_{i=1}^m c_i\phi_i, \sum_{j=1}^m c_j\phi_j\rangle = \\ &= \langle y,y\rangle - 2\sum_{i=1}^m c_i\langle y,\phi_i\rangle + \sum_{i=1}^m\sum_{j=1}^m c_ic_j\langle\phi_i,\phi_j\rangle \end{aligned}$$

É claro que D é uma função quadrática em $\mathbb{R}^m$ e portanto $D \in C^2(\mathbb{R}^m)$.

Começamos por determinar os pontos de estacionaridade de D. Trata-se dos pontos $c^* = (c_1^*,...,c_m^*) \in \mathbb{R}^m$ tais que $\nabla D(c^*) = 0$. Ora

$$\begin{gathered} \nabla D(c) = 0 \\ \Leftrightarrow \\ \frac{\partial D}{\partial c_i}(c) = 0, \quad i = 1,...,m \\ \Leftrightarrow \\ -2\langle y,\phi_i\rangle + 2\sum_{j=1}^m c_j\langle\phi_i,\phi_j\rangle = 0, \quad i = 1,...,m \\ \Leftrightarrow \\ \sum_{j=1}^m c_j\langle\phi_i,\phi_j\rangle = \langle y,\phi_i\rangle, \quad i = 1,...,m, \end{gathered}$$

logo os pontos de estacionaridade de D, se existirem, são solução do sistema linear

$$\sum_{j=1}^m c_j\langle\phi_i,\phi_j\rangle = \langle y,\phi_i\rangle, \quad i = 1,...,m$$

o qual, em notação matricial, se escreve

$$\begin{bmatrix} \langle\phi_1,\phi_1\rangle & \dots & \langle\phi_1,\phi_m\rangle \\ \vdots & \vdots & \vdots \\ \langle\phi_m,\phi_1\rangle & \dots & \langle\phi_m,\phi_m\rangle \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \langle y,\phi_1\rangle \\ \vdots \\ \langle y,\phi_m\rangle \end{bmatrix} \tag{9}$$

A este sistema chama-se *sistema normal* ou *sistema de equações normais*. Note-se que, devido à simetria dos produtos internos, $\langle\varphi,\psi\rangle = \langle\psi,\varphi\rangle$, a matriz do sistema (9) é simétrica.

Devemos agora estabelecer condições para que este sistema tenha uma e uma só solução, c^*, e para que um tal c^* seja ponto de mínimo (absoluto) de D. Para verificar se c^* é minimizante de D, calculamos a matriz Hessiana de D nesse ponto e verificamos se esta é definida positiva. Como

$$\frac{\partial^2 D}{\partial c_j \partial c_i}(c) = 2\langle \phi_i, \phi_j \rangle, \quad i, j = 1, ..., m$$

a matriz Hessiana de D é independente de c e dada por

$$H = 2 \begin{bmatrix} \langle \phi_1, \phi_1 \rangle & \dots & \langle \phi_1, \phi_m \rangle \\ \vdots & \vdots & \vdots \\ \langle \phi_m, \phi_1 \rangle & \dots & \langle \phi_m, \phi_m \rangle \end{bmatrix} = 2M$$

onde M designa a matriz do sistema normal (9).

Se M for definida positiva, isto é, se $\alpha^\top M \alpha > 0$, para todo $\alpha \in \mathbb{R}^m \setminus \{0\}$, então

(i) o sistema de equações normais terá uma e uma só solução $c^* = (c_1^*, ..., c_m^*)$, porque uma matriz definida positiva é não singular;

(ii) c^* será ponto de mínimo absoluto de D, pois, pela fórmula de Taylor de segunda ordem para campos escalares,

$$D(c) = D(c^*) + \nabla D(c^*) \cdot (c - c^*) + \frac{1}{2}(c - c^*)^\top H (c - c^*) = D(c^*) + (c - c^*)^\top M (c - c^*) > D(c^*)$$

para qualquer $c \in \mathbb{R}^m \setminus \{c^*\}$.

Assim, basta que a matriz M do sistema normal seja definida positiva para que D tenha um e um só ponto de mínimo. Para cada $\alpha = (\alpha_1, ..., \alpha_m) \in \mathbb{R}^m$ tem-se

$$\alpha^\top M \alpha = \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j M_{ij} = \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j \langle \phi_i, \phi_j \rangle$$

$$= \langle \sum_{i=1}^m \alpha_i \phi_i, \sum_{j=1}^m \alpha_j \phi_j \rangle = \left\| \sum_{i=1}^m \alpha_i \phi_i \right\|^2 \geq 0,$$

pelo que M é certamente semi-definida positiva, e portanto basta que M seja não singular para garantir que M é definida positiva.

Vejamos um critério para decidir se M é definida positiva em termos das funções de base ϕ_1,..., ϕ_m.

Teorema 3.1. *A matriz M é definida positiva se e só se as funções ϕ_1,...,ϕ_m são linearmente independentes*

(i) no conjunto $\{x_1, ..., x_n\}$, no caso discreto;

(ii) no intervalo $[a, b]$, no caso contínuo.

Demonstração. A matriz M é definida positiva se e só se

$$\left(\left\| \sum_{i=1}^{m} \alpha_i \phi_i \right\|^2 = 0 \right) \Rightarrow \alpha = 0. \tag{10}$$

No caso discreto, tem-se

$$\left\| \sum_{i=1}^{m} \alpha_i \phi_i \right\|^2 = \sum_{k=1}^{n} \left[\sum_{i=1}^{m} \alpha_i \phi_i(x_k) \right]^2$$

pelo que a condição (10) se escreve

$$\left(\sum_{k=1}^{n} \left[\sum_{i=1}^{m} \alpha_i \phi_i(x_k) \right]^2 = 0 \right) \Rightarrow \alpha = 0$$

ou ainda

$$\left(\sum_{i=1}^{m} \alpha_i \phi_i(x_k) = 0, \, \forall k \in \{1, ..., n\} \right) \Rightarrow \alpha = 0$$

o que significa que as funções ϕ_1,...,ϕ_m são linearmente independentes no conjunto $\{x_1, ..., x_n\}$.

No caso contínuo, (10) fica

$$\left(\int_a^b \left[\sum_{i=1}^{m} \alpha_i \phi_i(x) \right]^2 dx = 0 \right) \Rightarrow \alpha = 0,$$

ou seja,

$$\left(\sum_{i=1}^{m} \alpha_i \phi_i(x) = 0, \forall x \in [a, b] \right) \Rightarrow \alpha = 0,$$

o que significa que as funções ϕ_1,...,ϕ_m são linearmente independentes no intervalo $[a, b]$. □

Em resumo, o *algoritmo do método dos mínimos quadrados* consiste em determinar $c^* \in \mathbb{R}^m$ solução do *sistema normal*

$$\begin{bmatrix} \langle \phi_1, \phi_1 \rangle & \dots & \langle \phi_1, \phi_m \rangle \\ \vdots & \vdots & \vdots \\ \langle \phi_m, \phi_1 \rangle & \dots & \langle \phi_m, \phi_m \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \langle y, \phi_1 \rangle \\ \vdots \\ \langle y, \phi_m \rangle \end{bmatrix}$$

na hipótese que a matriz é não singular, ou de forma equivalente, que as funções de base $\phi_1, ..., \phi_m$ são linearmente independentes no conjunto de pontos de ajustamento.

No caso discreto, a matriz do sistema normal é da forma $M = B^\top B$, com

$$B = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_m(x_1) \\ \phi_1(x_2) & \dots & \phi_m(x_2) \\ \vdots & \dots & \vdots \\ \phi_1(x_n) & \dots & \phi_m(x_n) \end{bmatrix}$$

e

$$\begin{bmatrix} \langle y, \phi_1 \rangle \\ \vdots \\ \langle y, \phi_m \rangle \end{bmatrix} = B^\top \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

pelo que o *sistema normal* se pode escrever na forma

$$B^\top Bc = B^\top y \text{ com } B = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_m(x_1) \\ \phi_1(x_2) & \dots & \phi_m(x_2) \\ \vdots & \dots & \vdots \\ \phi_1(x_n) & \dots & \phi_m(x_n) \end{bmatrix} \text{ e } y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Exemplo 3.1. *A solubilidade de PuF_3 em $2Li-BeF_2$ fundido foi determinada a temperaturas entre 500 e 660^oC. Os valores obtidos estão registados na tabela seguinte, onde $x = 1000/\,T(K)$ e $y = -\log_{10}$(solubilidade/%mol)*

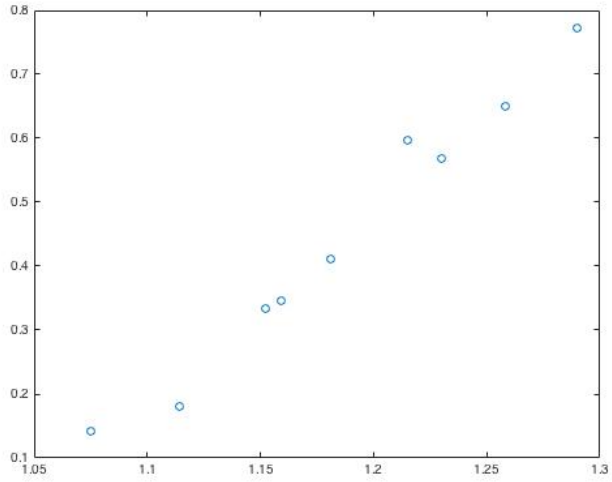


Figura 4: Representação gráfica dos pontos tabelados

x	y
1.075	*0.141*
1.114	*0.181*
1.152	*0.333*
1.159	*0.346*
1.181	*0.412*
1.215	*0.597*
1.230	*0.569*
1.258	*0.650*
1.290	*0.772*

A representação gráfica dos pontos tabelados (Fig. 4) sugere a existência de uma relação linear ($y = c_1 + c_2 x$) *entre* x *e* y. *Assim, a função de ajustamento será da forma*

$$g(x) = c_1\phi_1(x) + c_2\phi_2(x),$$

com $\phi_1(x) = 1$ *e* $\phi_2(x) = x$ *e portanto, o sistema normal fica*

$$\begin{bmatrix} 9 & \sum_{i=1}^{9} x_i \\ \sum_{i=1}^{9} x_i & \sum_{i=1}^{9} x_i^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{9} y_i \\ \sum_{i=1}^{9} x_i y_i \end{bmatrix}$$

A matriz deste sistema é não singular e, consequentemente, a solução $c_1 = -3.21674$, $c_2 = 3.087184$ *é minimizante da função*

$$D(c_1, c_2) := \sum_{k=1}^{9} [y_k - (c_1 + c_2 x_k)]^2$$

Conclui-se que a reta que melhor se ajusta aos dados, no sentido dos mínimos quadrados, é $y(x) = -3.21674 + 3.087184x$.

Exemplo 3.2. *Pretende-se aproximar a função* $y(x) = \cos x$, *no intervalo* $[0, 1]$, *por uma função do tipo* $g(x) = a + bx^2 + cx^4$, *de modo que*

$$\int_0^1 [y(x) - g(x)]^2 \, dx$$

seja mínimo. As funções de base são $\phi_1(x) = 1$, $\phi_2(x) = x^2$ *e* $\phi_3(x) = x^4$, *logo o sistema normal é*

$$\begin{bmatrix} 1 & 0.333333 & 0.2 \\ 0.333333 & 0.333333 & 0.142857 \\ 0.333333 & 0.142857 & 0.111111 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0.63662 \\ 0.120595 \\ 0.0501149 \end{bmatrix}$$

e a função de ajustamento $g(x) = 0.999971 - 0.499385x^2 + 0.0398087x^4$. *Note-se que os gráficos de* y *e* g *ficam praticamente sobrepostos.*

Exemplo 3.3. *Considere-se a tabela de pontos*

x_i	-1	0	1
f_i	2	0	1

à qual se pretende ajustar, segundo o critério dos mínimos quadrados, uma função do tipo $g(x) = c_1 x + c_2 \sin\left(\frac{\pi}{2}x\right)$. *O sistema normal é*

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

o qual não tem solução. Isto deve-se ao facto de as funções de base escolhidas não serem linearmente independentes no conjunto $\{-1, 0, 1\}$.

A implementação do método dos mínimos quadrados é feita no programa MATLAB seguinte, o qual generaliza a função `polyfit` para ajustamentos polinomiais:

```
function g = minimos_quadrados(x,y,fb)
% Recebe a tabela de pontos a ajustar (x,y) e um conjunto de funções de base fb,
% retorna a função g que melhor ajusta os dados no sentido dos mínimos quadrados
% x e y são vetores linha
n=length(x);
m=length(fb);
B=zeros(n,m);
for j=1:m
    B(:,j)=fb{j}(x)';
end;
%Resolução do sistema normal
c=B'*B\B'*y';
%Construção da função de ajustamento
g = @(t)0;
for i=1:m
    g=@(t)(c(i)*fb{i}(t)+g(t));
end;
end
```

3.2 Aplicação de regressão linear para determinação da ordem de convergência de um método iterativo para equações não-lineares

Vamos ver como se pode determinar experimentalmente a ordem e coeficiente assintótico de convergência de um método iterativo. Partindo da definição

$$\lim_{n\to\infty} \frac{|z - x_{n+1}|}{|z - x_n|^p} = K_\infty$$

usa-se a relação assintótica (i.e, para n suficientemente grande)

$$|z - x_{n+1}| \approx K_\infty |z - x_n|^p,$$

a qual conduz a

$$\log(|z - x_{n+1}|) \approx \log(K_\infty) + p \log(|z - x_n|). \tag{11}$$

Isto sugere uma relação linear/afim entre $\log(|z-x_{n+1}|)$ e $\log(|z-x_n|)$ à qual se pode aplicar regressão linear para estimar p e K_∞.

Exemplo 3.4. *Considere-se a equação* $\exp(-x) - \sin(x) = 0$ *com raiz* $z \in [0, 1]$. *O comando* ***fzero*** *do* MATLAB *permite obter imediatamente*

```
 >> fzero(@(x)exp(-x)-sin(x),[0,1])
ans = 0.588532743981861
```

Aplica-se agora o método de Newton (implementado numa função MATLAB `Newton_Method`*) à equação anterior, de modo a aproximar z, tomando $x_0 = -2$ como iterada inicial. A tolerância de erro imposta é 0.5×10^{-16}. As estimativas de erros resultam da diferença entre duas iteradas consecutivas.*

```
>> s=Newton_Method( @(x)exp(-x)-sin(x), -2, 0.5*10^-16, 20 )
s =
        -0.809915171200757          1.19008482879924
         0.201887872725063          1.01180304392582
         0.545076398239476         0.343188525514412
         0.587807390439059        0.0427309921995831
         0.588532533529297      0.000725143090238412
         0.588532743981843      2.10452546389739e-07
         0.588532743981861        1.765254609154e-14
         0.588532743981861                         0
```

A primeira coluna contém as iteradas do método de Newton e na segunda estão registadas as correspondentes estimativas de erro. Dada a precisão do resultado obtido, tomamos $z \approx 0.588532743981861$ e calculamos os erros das iteradas obtidas relativamente a este valor:

```
>> erros=0.588532743981861-s(:,1)
erros =
         1.39844791518262
        0.386644871256798
       0.0434563457423854
     0.000725353542802343
     2.10452563931263e-07
     1.75415237890775e-14
    -1.11022302462516e-16
    -1.11022302462516e-16
```

Usando estes valores para os erros e a relação assintótica (11)*, vamos estimar a ordem e coeficiente assintótico de convergência do método para este caso.*

```
>> polyfit(log(abs(erros(1:5))),log(abs(erros(2:6))),1)
ans = 1.96689014827941          -1.3056791986773

>> exp(-1.3056791986773)
```

```
ans = 0.270988417551734
```

Conclui-se que $p \approx 1.96689014827941 \approx 2$ *e* $K_\infty \approx 0.270988417551734$*. O resultado obtido para a ordem de convergência está de acordo com o esperado para o método de Newton, ou seja,* $p = 2$*, mas usando a informação sobre a ordem de convergência, temos, tomando* $z \approx 0.588532743981861$*:*

$$\lim_{n\to\infty} \frac{|z - x_{n+1}|}{|z - x_n|^2} = \frac{|f''(z)|}{2|f'(z)|} \approx 0.4002756438256682$$

ou seja, seria de esperar $K_\infty \approx 0.4002756438256682$ *para o coeficiente assintótico de convergência. Recorrendo novamente às estimativas de erro, e usando*

$$\lim_{n\to\infty} \frac{|z - x_{n+1}|}{|z - x_n|^2} = K_\infty$$

obtém-se

```
>> abs(erros(2:6))./erros(1:5).^2
ans =

          0.19770591467625
         0.290689065712319
         0.384099135300946
         0.399995170673378
          0.39605807706668
```

o que está mais de acordo com

$$\lim_{n\to\infty} \frac{|z - x_{n+1}|}{|z - x_n|^2} = \frac{|f''(z)|}{2|f'(z)|}$$

e nos permite considerar $K_\infty \approx 0.40$*.*

4 Recursos computacionais

Exemplo 4.1. *Os dados representados na tabela*

Dia	0	2	4	6	8	10	12	14	16	18	20
Qt. $\times 10^{-6}$	67.38	70.93	74.67	78.60	82.74	87.10	91.69	92.51	101.60	106.95	112.58

referem-se ao crescimento de uma bactéria num meio de cultura líquido ao longo de vários dias. Pretende-se obter previsões da quantidade de bactérias ao fim de ao fim de 22, 25 e 30 dias, e para tal iremos recorrer ao polinómio interpolador dos pontos tabelados e à reta de regressão.

No MATHEMATICA a função `Fit` permite fazer o ajustamento de dados discretos através de interpolação e melhor aproximação mínimos quadrados. Para o Exemplo 4.1:

$$\begin{aligned}
&\texttt{In}: \text{dados} = \{\{0, 67.38\}, \{2, 70.93\}, \{4, 74.67\}, \{6, 78.60\}, \{8, 82.74\},\\
&\qquad \{10, 87.10\}, \{12, 91.69\}, \{14, 92.51\}, \{16, 101.60\},\\
&\qquad \{18, 106.95\}, \{20, 112.58\}\}\\
&\texttt{In}: \text{reta} = \text{Fit}[\text{dados}, \{1, x\}, x]\\
&\texttt{Out}: \quad 65.7209 + 2.21655x\\
&\texttt{In}: \text{polint} = \text{Fit}[\text{dados}, \{1, x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}\}, x]\\
&\texttt{Out}: \quad 67.38 - 32.6639x + 47.9288x^2 - 26.7905x^3 + 8.04624x^4 - 1.44855x^5\\
&\qquad +0.163585x^6 - 0.0116686x^7 + 0.000509742x^8 + 0.0000124324x^9 + 1.29501 \times 10^{-7}x^{10}
\end{aligned}$$

O MATHEMATICA dispõe ainda da função `InterpolatingPolynomial`. Neste caso, poderíamos ter feito:

$$\texttt{In}: \quad \text{polint} = \text{InterpolatingPolynomial}[\text{dados}, x]$$

que teria produzido

$$\begin{aligned}
&112.58 + (-20 + x)(2.26 + (0.0288 + (0.000242708\\
&+(7.8125 * 10^{-7} + (-3.28621 * 10^{-7} + (2.17014 * 10^{-8}\\
&+(-9.81729 * 10^{-9} + (2.07162 * 10^{-6} + (2.58694 * 10^{-7}\\
&+1.29501 * 10^{-7}(-6 + x))(-14 + x))(-8 + x))(-18\\
&+x))(-2 + x))(-16 + x))(-4 + x))(-10 + x))x)
\end{aligned}$$

No MATLAB a função `polyfit` permite fazer o ajustamento de dados discretos através de interpolação polinomial e melhor aproximação mínimos quadrados.

A informação disponível sobre este comando, que pode ser obtida recorrendo ao `help`, é a seguinte:

" P = polyfit(X,Y,N) finds the coefficients of a polynomial P(X) of degree N that fits the data Y best in a least-squares sense. P is a row vector of length N+1 containing the

polynomial coefficients in descending powers,

$$P(1) * X^N + P(2) * X^{N-1} + ... + P(N) * X + P(N+1).''$$

A seguir mostra-se uma aplicação ao Exemplo 4.1.

```
>> dados = [0 67.38;2 70.93;4 74.67;6 78.60;8 82.74;10 87.10;12 91.69;
                                  14 92.51;16 101.60;18 106.95;20 112.58];

>> polyfit(dados(:,1),dados(:,2),1)
ans = 2.21654545454546          65.7209090909091

>> polyfit(dados(:,1),dados(:,2),10)
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT. > In polyfit (line 75)

ans =
  Columns 1 through 3
1.29500562332915e-07 -1.24323607744162e-05 0.000509742050539577
  Columns 4 through 6
0.0116685519740109  0.163584972680757 -1.44854804134423
  Columns 7 through 9
8.04624460850376 -26.7904611628681 47.9287553539012
  Columns 10 through 11
-32.6639166657563    67.3800000025109
```

Note-se que o sistema linear que permite obter o polinómio interpolador p_{10} é resolvido pelo MATLAB, mas é detetado o mau condicionamento do sistema, sendo emitido um aviso a dar conta desse problema.

Ainda relativamente ao Exemplo 4.1, criámos o script que se segue, o qual permite resolver o problema de interpolação.

```
% Dados do problema
x = linspace(0,20,11);
y = 1e6*[67.38 70.93 74.67 78.60 82.74 87.10 91.69 92.51 101.60 106.95 112.58];
% Polinómio interpolador de Lagrange
pint = polyfit(x,y,10);
% Reta de regressão
```

```
preg = polyfit(x,y,1);
% Representação gráfica dos dados e das funções de ajustamento
x1 = linspace(0,20);
y1 = polyval(pint,x1);
y2 = polyval(preg,x1);
figure
plot(x,y,'o')
hold on
plot(x1,y1)
hold on
plot(x1,y2)
hold off
xlabel('Tempo (dias)');
ylabel('Quantidade de bacterias');
legend('Dados','Interpol. polinomial','Regressao linear');
% Previsões apresentadas numa tabela
d=[22 25 30];ydint=polyval(pint,d);ydreg=polyval(preg,d);
table(d',ydint',ydreg','VariableNames',{'Dias','Bacterias_Int','Bacterias_Reg'})
```

A execução produziu uma tabela com resultados

```
>> interpolation
Warning: Polynomial is badly conditioned. Add points with distinct X
values, reduce the degree of the polynomial, or try centering and
scaling as described in HELP POLYFIT.
> In polyfit (line 75)
  In interpolation (line 5)

ans =

    Dias    Bacterias_Int    Bacterias_Reg
    ____    _____________    _____________

    22      1.4416e+09       1.1448e+08
    25      3.1148e+10       1.2113e+08
    30      9.0322e+11       1.3222e+08
```

e o gráfico da Figura 5. Note-se a mensagem de aviso alertando para o mau condicionamento do problema de interpolação polinomial.

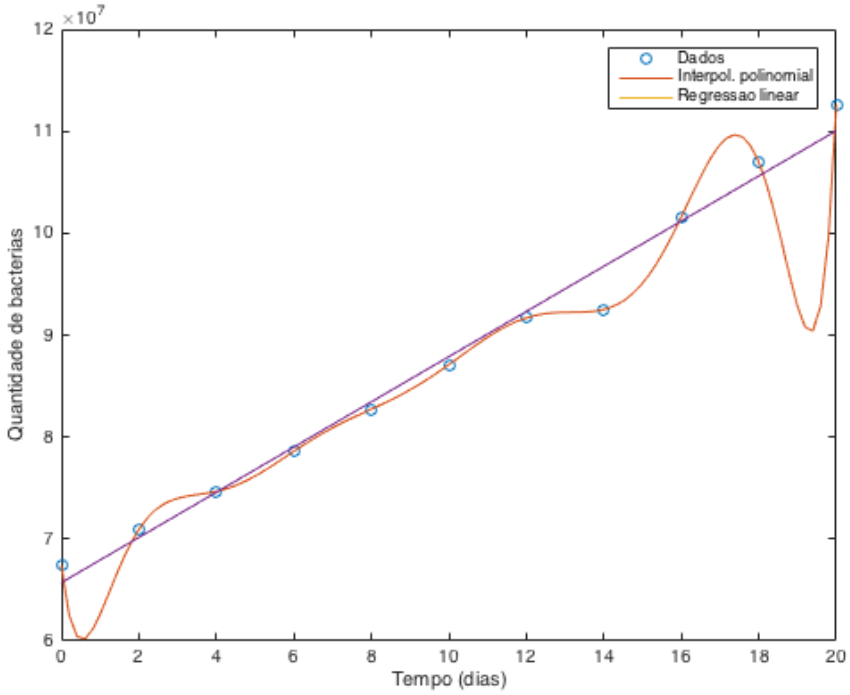


Figura 5: Funções de ajustamento da tabela referente ao crescimento do número de bactérias

Referências

[1] M. Atteia e M. Pradel, Elements d'Analyse Numérique, Cepadues-Editions, 1990.

[2] W. Gander, Change of basis in polynomial interpolation, *Numer. Linear Algebra Appl.* 2000; 00:1-6.

[3] R. Kress, Numerical Analysis, Springer-Verlag, 1998.

[4] A. Quarteroni, R. Sacco e F. Saleri, Cálculo Científico com Matlab e Octave, Springer-Verlag, 2007 (traduzido por Adélia Sequeira).

[5] F. Romeiras, Matemática Computacional, Apontamentos das aulas, 2008.