Matemática numérica

Pedro H A Konzen

29 de maio de 2018

Licença

Este trabalho está licenciado sob a Licença Atribuição-CompartilhaIgual 4.0 Internacional Creative Commons. Para visualizar uma cópia desta licença, visite http://creativecommons.org/licenses/by-sa/4.0/deed.pt_BR ou mande uma carta para Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Prefácio

Nestas notas de aula são abordados temas introdutórios de matemática numérica. Como ferramenta computacional de apoio didático, faço uso de códigos em `GNU Octave` (compatíveis com `MATLAB`). Ressalto que os códigos apresentados são implementações ingênuas com intuito de ressaltar os aspectos fundamentais dos métodos numéricos discutidos no texto.
Agradeço aos(às) estudantes que assiduamente ou esporadicamente contribuem com correções, sugestões e críticas em prol do desenvolvimento deste material didático.

Pedro H A Konzen

Sumário

Capítulo 1

Introdução

Em construção ...

Licença CC-BY-SA 4.0

Capítulo 2

Aproximação por mínimos quadrados

2.1 Problemas lineares

Dado um conjunto de n pontos $\{(x_i,y_i)\}_{i=1}^n$, $x_i \neq x_j$ para $i \neq j$, e uma família de $m \leq n$ funções $\{f_i(x)\}_{i=1}^m$, o problema linear de aproximação por mínimos quadrados consiste em determinar os m coeficientes $\{c_i\}_{i=1}^m$ tal que a função

$$f(x;c) = \sum_{j=1}^{m} c_j f_j(x) \tag{2.1}$$

$$= c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x) + \cdots + c_m f_m(x) \tag{2.2}$$

aproxime o conjunto de pontos dados no sentido de mínimos quadrados, i.e. o vetor dos coeficientes $c = (c_1, c_2, \ldots, c_m)$ é solução do seguinte problema linear de minimização

$$\min_c \left\{ E := \sum_{i=1}^{n} (y_i - f(x_i;c))^2 \right\}. \tag{2.3}$$

A fim de trabalharmos com uma notação mais compacta, definimos o resíduo $r(c) = (r_1(c), r_2(c), \ldots, r_n(c))$, onde $r_i(c) := y_i - f(x_i)$ e $c = (c_1, c_2, \ldots, c_m)$. Com esta notação, o problema de mínimos quadrados se resume a resolver

$$\min_c \{E := \|r(c)\|_2^2\}. \tag{2.4}$$

Licença CC-BY-SA 4.0

2.1.1 Método das equações normais

A fim de resolver o problema de mínimos quadrados (2.4), observamos que o erro quadrático

$$
\begin{aligned}
E &= \|r(c)\|_2^2 && (2.5)\\
&= \sum_{i=1}^{n} r_i(c)^2 && (2.6)\\
&= \sum_{i=1}^{n} \left(y_i - f(x_i; c)\right)^2 && (2.7)\\
&= \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} c_j f_j(x_i)\right)^2 && (2.8)\\
&= \|y - Ac\|_2^2, && (2.9)
\end{aligned}
$$

onde $y = (y_1, y_2, \ldots, y_n)$ e

$$
A := \begin{bmatrix}
f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\
f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\
\vdots & \vdots & \vdots & \vdots \\
f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n)
\end{bmatrix}. \tag{2.10}
$$

Os parâmetros c_j que minimizam o erro E são solução do seguinte sistema de equações

$$
\frac{\partial E}{\partial c_j} = 2 \sum_{i=0}^{n} r_i(c) \frac{\partial}{\partial c_j} r_i(c) = 0, \tag{2.11}
$$

onde $j = 1, 2, \ldots, m$. Ou, em uma notação mais apropriada,

$$
\begin{aligned}
\nabla_c E = 0 &\Leftrightarrow A^T r(c) = 0 && (2.12)\\
&\Leftrightarrow A^T (y - Ac) = 0 && (2.13)\\
&\Leftrightarrow A^T A c = A^T y. && (2.14)
\end{aligned}
$$

Portanto, o problema linear de mínimos quadrados se resume em resolver as chamadas **equações normais**

$$
A^T A c = A^T y. \tag{2.15}
$$

Licença CC-BY-SA 4.0

Logo, o problema linear de mínimos quadrados (2.4) reduz-se a resolver o sistema linear (2.15) para c. Isto nos leva a questão de verificar se A^TA é invertível. De sorte, da disciplina de álgebra linear temos o seguinte teorema.

Teorema 2.1.1. *A matriz A^TA é positiva definida se, e somente se, as colunas de A são linearmente independentes (i.e. posto$(A) = n$).*

Demonstração. Se as colunas de A são linearmente independentes, então $x \neq 0$ implica $Ax \neq 0$ e, equivalentemente, $x^TA^T \neq 0$. Portanto, $x \neq 0$ implica $x^TA^TAx = \|Ax\|_2^2 > 0$, o que mostra que A^TA é positiva definida.
Suponhamos, agora, que as colunas de A não são linearmente independentes. Então, existe $x_0 \neq 0$ tal que $Ax_0 = 0$. Mas, então, $x_0^TA^TAx_0 = 0$, o que mostra que A^TA não é positiva definida. □

Este teorema nos fornece uma condição suficiente para a existência (e unicidade) de solução do problema linear de mínimos quadrados. Mais especificamente, se as colunas da matriz A são linearmente independentes, então os coeficientes da função $f(x)$ que melhor ajustam os pontos dados são

$$c = (A^TA)^{-1}A^Ty. \tag{2.16}$$

Exemplo 2.1.1. (Ajuste de polinômios) Considere o problema de ajustar o conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	1,2	$-0,1$	0,7	2,4

por um polinômio quadrático da forma

$$p(x) = p_1x^2 + p_2x + p_n \tag{2.17}$$

no sentido de mínimos quadrados.
Neste caso, a família de funções do problema de mínimos quadrados é $f_1(x) = x^2$, $f_2(x) = x$ e $f_3(x) = 1$. Assim sendo, os coeficientes $p = (p_1, p_2, p_3)$ são solução do seguinte sistema linear

$$A^TAp = A^Ty, \tag{2.18}$$

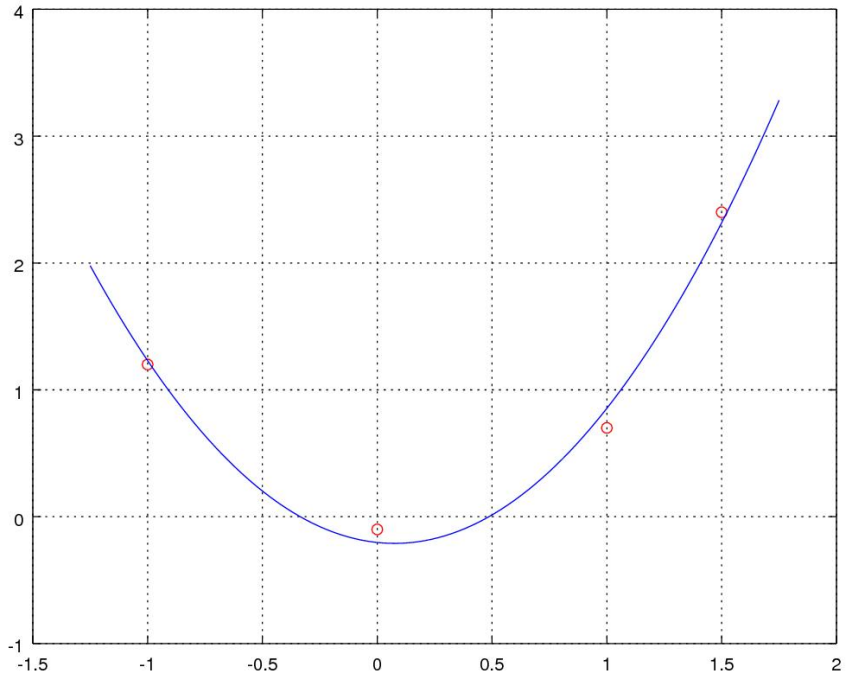

Figura 2.1: Esboço do polinômio ajustado no Exemplo 2.1.1.

onde $y = (y_1, y_2, y_3)$ e

$$A := \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{bmatrix}. \tag{2.19}$$

Emfim, resolvendo as equações normais (2.18), obtemos

$$p(x) = 1{,}25x^2 - 0{,}188x - 0{,}203. \tag{2.20}$$

A Figura 2.2 mostra um esboço dos pontos (em vermelho) e do polinômio ajustado (em azul).
Os coeficientes e um esboço do polinômio ajustado podem ser obtidos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [1.2, -0.1, 0.7, 2.4]';

#resol. as eqs. normais
A = [x.^2 x.^1 x.^0];
p = inv(A'*A)*A'*y

#esboco do pol. ajustado
xx = linspace(-1.25,1.75);
plot(x,y,'ro',...
     xx,polyval(p,xx));grid
```

Exemplo 2.1.2. (Ajuste de curvas) Consideremos o mesmo conjunto de pontos do exemplo anterior (Exemplo 2.1.1). Aqui, vamos ajustar uma curva da forma

$$f(x) = c_1 \operatorname{sen}(x) + c_2 \cos(x) + c_3 \tag{2.21}$$

no sentido de mínimos quadrados. Para tanto, formamos a matriz

$$A := \begin{bmatrix} \operatorname{sen}(x_1) & \cos(x_1) & 1 \\ \operatorname{sen}(x_2) & \cos(x_2) & 1 \\ \operatorname{sen}(x_3) & \cos(x_3) & 1 \\ \operatorname{sen}(x_4) & \cos(x_4) & 1 \end{bmatrix} \tag{2.22}$$

e, então, resolvemos as equações normais $A^T Ac = A^T y$ para o vetor de coeficientes $c = (c_1, c_2)$. Fazendo isso, obtemos $c_1 = -0{,}198$, $c_2 = -2{,}906$ e $c_3 = 2{,}662$. A Figura **??** mostra um esboço da curva ajustada (linha azul) aos pontos dados (círculos vermelhos).
Os coeficientes e um esboço do polinômio ajustado podem ser obtidos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [1.2, -0.1, 0.7, 2.4]';

#resol. as eqs. normais
A = [sin(x) cos(x) ones(4,1)];
```

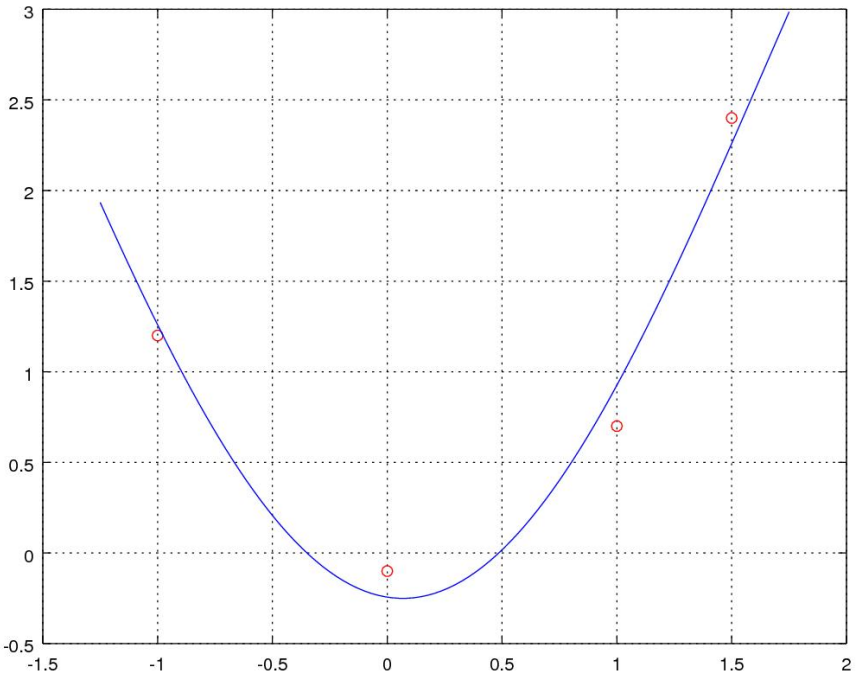

Figura 2.2: Esboço da curva ajustada no Exemplo 2.1.2.

```
c = inv(A'*A)*A'*y

#curva ajustada
f = @(x) c(1)*sin(x) + c(2)*cos(x) + c(3)

#esboco da fun. ajustada
xx = linspace(-1.25,1.75);
plot(x,y,'ro',...
     xx,f(xx));grid
```

Exemplo 2.1.3. (Um problema não linear) Consideremos o problema de ajustar, no sentido de mínimos quadrados, à função

$$f(x) = c_1 e^{c_2 x} \tag{2.23}$$

ao seguinte conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	8,0	1,5	0,2	0,1

Aqui, temos um problema não linear de mínimos quadrados que pode ser transformado em um problema linear fazendo-se

$$y = c_1 e^{c_2 x} \Rightarrow \ln y = \ln c_1 e^{c_2 x} \tag{2.24}$$

$$\Rightarrow \ln y = \ln c_1 + c_2 x. \tag{2.25}$$

Isto é, denotando $d_1 := \ln c_1$ e $d_2 := c_2$, o problema se resume a ajustar uma reta $r(x) = d_1 + d_2 x$ ao conjunto de pontos $\{(x_i, \ln y_i)\}_{i=1}^4$.
Para resolver o problema transformado, formamos a matriz

$$A := \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} \tag{2.26}$$

e, então, resolvemos as equações normais $A^T A d = A^T \ln y$, com $\ln y = (\ln y_1, \ln y_2, \ln y_3, \ln y_4)$, donde obtemos $d_1 = 0{,}315$ e $d_2 = -1{,}792$. Das definições de d_1 e d_2, temos $c_2 = d_2 = -1{,}792$ e $c_1 = e^{d_1} = 1{,}371$. A Figura 2.3 mostra um esboço da curva $f(x) = c_1 e^{c_2 x}$ ajustada (linha azul) aos pontos dados (círculos vermelhos).
O ajuste e um esboço da função ajustada podem ser feitos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#resol. as eqs. normais
A = [ones(4,1) x];
d = inv(A'*A)*A'*log(y)

#fun. ajustada
```

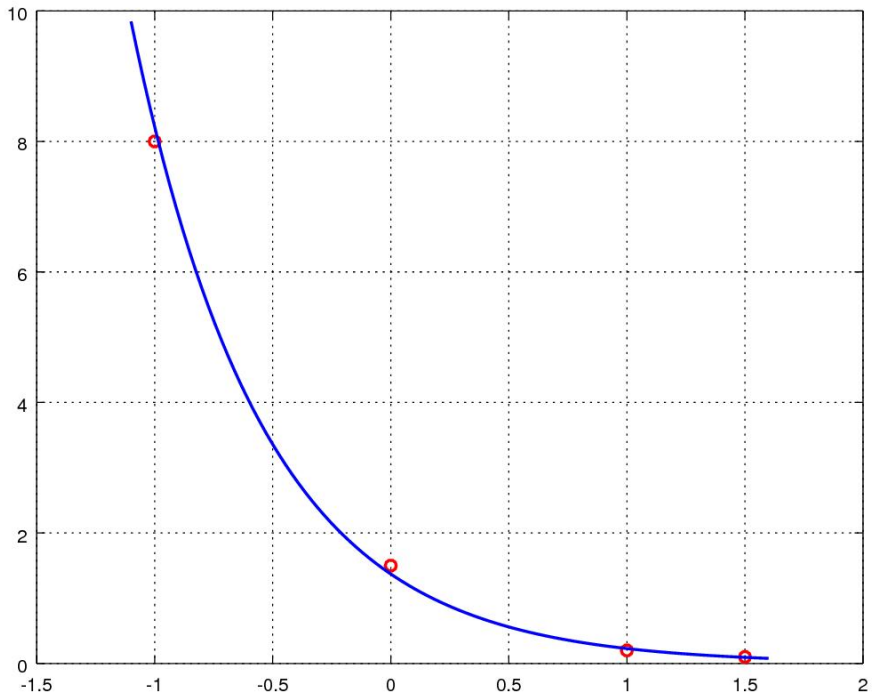

Figura 2.3: Esboço da curva ajustada no Exemplo 2.1.3.

```
c = [exp(d(1)); d(2)]
f = @(x) c(1)*exp(c(2)*x);

#esboco da fun. ajustada
xx = linspace(-1.1,1.6);
plot(x,y,'ro','linewidth',1.5,...
     xx,f(xx),'b-','linewidth',1.5);grid
```

Exercícios

E 2.1.1. Determine a reta $y = c_1x + c_2$ que melhor se ajusta, no sentido de mínimos quadrados, aos pontos

i	1	2	3	4	5
x_i	$-2{,}5$	$-1{,}3$	$0{,}2$	$1{,}7$	$2{,}3$
y_i	$3{,}8$	$1{,}5$	$-0{,}7$	$-1{,}5$	$-3{,}2$

Por fim, compute a norma L^2 do resídulo, i.e. $\|r(c)\|_2 = \|y - (c_1 x - c_2)\|_2$ para os pontos dados.

E 2.1.2. Determine o polinômio $y = c_1 x^3 + c_2 x^2 + c_3 x + c_4$ que melhor se ajusta, no sentido de mínimos quadrados, aos pontos

i	1	2	3	4	5
x_i	$-2{,}5$	$-1{,}3$	$0{,}2$	$1{,}7$	$2{,}3$
y_i	$3{,}8$	$0{,}5$	$2{,}7$	$1{,}2$	$-1{,}3$

Por fim, compute a norma L^2 do resídulo, i.e. $\|r(c)\|_2$.
Em construção ...

2.2 Problemas não lineares

Um problema não-linear de mínimos quadrados consiste em ajustar uma dada função $f(x; c)$ que dependa não linearmente dos parâmetros $c = (c_1, c_2, \ldots, c_m)$, $m \geq 1$, a um dado conjunto de $n \geq m$ pontos $\{(x_i, y_i)\}_{i=1}^n$. Mais especificamente, buscamos resolver o seguinte problema de minimização

$$\min_{\{c_1, c_2, \ldots, c_m\}} \left[E := \sum_{i=1}^{n} \left(y_i - f(x_i; c) \right)^2 \right]. \tag{2.27}$$

Aqui, denotaremos por $r(c) := (r_1(c), r_2(c), \ldots, r_n(c))$ o vetor dos resíduos $r_i(c) := y_i - f(x_i, c)$. Com isso, o problema se resume a encontrar o vetor de parâmetros c que minimiza

$$E = \|r(c)\|_2^2. \tag{2.28}$$

Tais parâmetros são solução do seguinte sistema de equações

$$\frac{\partial E}{\partial c_j} = 2 \sum_{i=1}^{n} r_i(c) \frac{\partial}{\partial c_j} r_i(c) \tag{2.29}$$

Licença CC-BY-SA 4.0

ou, equivalentemente, da equação

$$\nabla E = 0 \Leftrightarrow J_R^T(c)r(c) = 0, \tag{2.30}$$

onde

$$J_R(c) := \begin{bmatrix} \frac{\partial r_1}{\partial c_1} & \frac{\partial r_1}{\partial c_2} & \cdots & \frac{\partial r_1}{\partial c_m} \\ \frac{\partial r_2}{\partial c_1} & \frac{\partial r_2}{\partial c_2} & \cdots & \frac{\partial r_2}{\partial c_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_n}{\partial c_1} & \frac{\partial r_n}{\partial c_2} & \cdots & \frac{\partial r_n}{\partial c_m} \end{bmatrix} \tag{2.31}$$

é a jacobiana do resíduo r em relação aos parâmetros c.
Podemos usar o método de Newton para resolver (2.30). Para tanto, escolhemos uma aproximação inicial para $c^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_m^{(1)})$ e iteramos

$$H_R(c^{(k)})\delta^{(k)} = -J_R^T(c)r(c) \tag{2.32}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}, \tag{2.33}$$

onde $\delta^{(k)} = (\delta_1^{(k)}, \delta_2^{(k)}, \delta_m^{(k)})$ é a atualização de Newton (ou direção de busca) e $H_R(c) := [h_{p,q}(c)]_{p,q=1}^{m,m}$ é a matrix hessiana, cujos elementos são

$$h_{p,q} := \sum_{i=1}^{n} \left\{ \frac{\partial r_i}{\partial c_q}\frac{\partial r_i}{\partial c_p} + r_i \frac{\partial^2 r_i}{\partial c_q \partial c_p} \right\}. \tag{2.34}$$

Exemplo 2.2.1. Consideremos o problema de ajustar, no sentido de mínimos quadrados, a função

$$f(x; c) = c_1 e^{c_2 x} \tag{2.35}$$

ao seguinte conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	8,0	1,5	0,2	0,1

Aqui, vamos utilizar a iteração de Newton para o problema de mínimos quadrados, i.e. a iteração dada em (2.32)-(2.33). Para tanto, para cada

$i = 1, 2, 3, 4$, precisamos das seguintes derivadas parciais do resíduo $r_i(c) := y_i - c_1 e^{c_2 x_i}$:

$$\frac{\partial}{\partial c_1} r_i(c) = -e^{c_2 x_i}, \tag{2.36}$$

$$\frac{\partial}{\partial c_2} r_i(c) = -c_1 x_i e^{c_2 x_i}, \tag{2.37}$$

$$\frac{\partial^2}{\partial c_1^2} r_i(c) = 0, \tag{2.38}$$

$$\frac{\partial^2}{\partial c_1 \partial c_2} r_i(c) = \frac{\partial^2}{\partial c_2 \partial c_1} r_i(c) = -x_i e^{c_2 x_i}, \tag{2.39}$$

$$\frac{\partial^2}{\partial c_2^2} r_i(c) = -c_1 x_i^2 e^{c_2 x_i}. \tag{2.40}$$

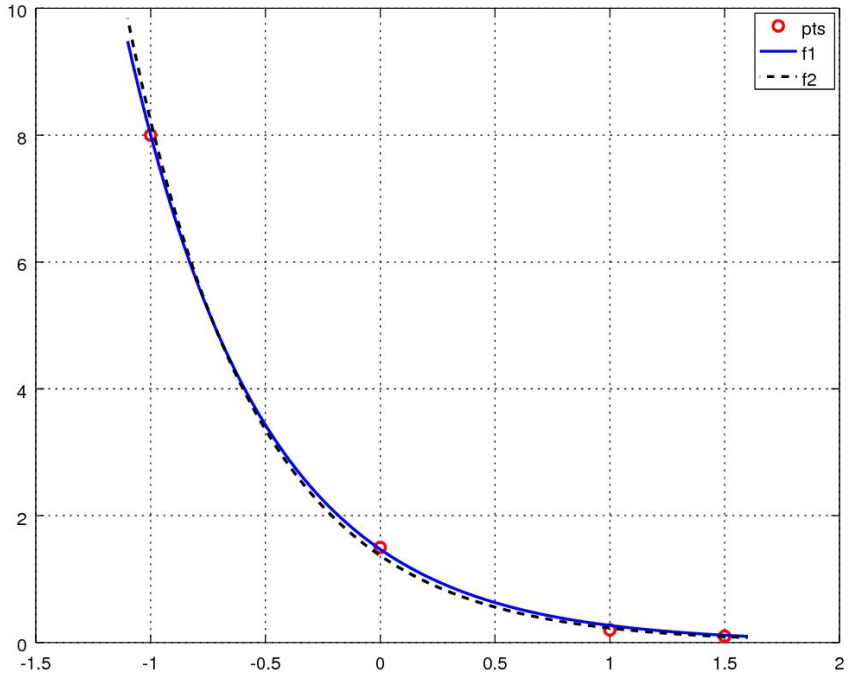

Figura 2.4: Esboço da curva ajustada no Exemplo 2.2.1.

Com isso e tomando $c^{(1)} = (1{,}4,\ -1{,}8)$ (motivado do Exemplo 2.1.3), computamos as iterações de Newton (2.32)-(2.33). Iterando até a precisão de $TOL = 10^{-4}$, obtemos a solução $c_1 = 1{,}471$ e $c_2 = -1{,}6938$. Na Figura 2.4 vemos uma comparação entre a curva aqui ajustada $(-)$ e aquela obtida no Exemplo 2.1.3 $(--)$.
O ajuste discutido neste exemplo pode ser computado no `GNU Octave` com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
global y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);

#residuo
r = @(c) y - f(x,c);

#jacobiana
function A = J(c)
  global x
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
endfunction

#hessiana
function A = H(c)
  global x
  global y
  A = zeros(2,2);
  A = J(c)'*J(c);
  for i=1:4
    A(1,1) += 0;
    A(1,2) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- x(i)*exp(c(2)*x(i)));
    A(2,1) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- x(i)*exp(c(2)*x(i)));
```

```
    A(2,2) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- c(1)*x(i)^2*exp(c(2)*x(i)));
  endfor
endfunction

#aprox. inicial
c = [1.4 -1.8]';

#iteracoes de Newton
k=0;
do
  k+=1;
  delta = - inv(H(c))*J(c)'*r(c);
  c = c + delta;
  [k,c',norm(delta)]
until ((k>10) | (norm(delta)<1e-4))
```

Observamos que a solução obtida no exemplo anterior (Exemplo 2.2.1) difere da previamente encontrada no Exemplo 2.1.3. Naquele exemplo, os parâmetros obtidos nos fornecem $E = 6{,}8\text{E}{-2}$, enquando que a solução do exemplo anterior fornece $E = 6{,}1\text{E}{-3}$. Isto é esperado, pois naquele exemplo resolvemos um problema aproximado, equanto no exemplo anterior resolvemos o problema por si.
O emprego do método de Newton para o problema de mínimos quadrados tem a vantagem da taxa de convergência quadrática, entretanto requer a computação das derivadas parciais de segunda ordem do resíduo. Na sequência discutimos alteranativas comumente empregadas.

2.2.1 Método de Gauss-Newton

O método de Gauss-Newton é uma técnica iterativa que aproxima o problema não linear de mínimos quadrados (2.27) por uma sequência de problemas lineares. Para seu desenvolvimento, começamos de uma aproximação inicial $c^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_m^{(1)})$ dos parâmetros que queremos ajustar. Também, assumindo que a n-ésima iterada $c^{(k)}$ é conhecida, faremos uso da aproximação de primeira ordem de $f(x,c)$ por polinômio de Taylor, i.e.

$$f(x; c^{(k+1)}) \approx f(x; c^{(k)}) + \nabla_c f(x; c^{(k)})(c^{(k+1)} - c^{(k)}), \tag{2.41}$$

onde

$$\nabla_c f(x;c) = \left[\frac{\partial}{\partial c_1} f(x;c) \ \frac{\partial}{\partial c_2} f(x;c) \ \cdots \ \frac{\partial}{\partial c_m} f(x;c) \right]. \tag{2.42}$$

O método consiste em obter a solução do problema não linear (2.27) pelo limite dos seguintes problemas lineares de mínimos quadrados

$$\min_{\delta^{(k)}} \left[\tilde{E} := \sum_{i=1}^{n} (y_i - f(x_i, c^{(k)}) - \nabla_c f(x_i; c^{(k)}) \delta^{(k)})^2 \right] \tag{2.43}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{2.44}$$

Agora, usando a notação de resíduo $r(c) = y - f(x;c)$, observamos que (2.50) consiste no problema linear de mínimos quadrados

$$\min_{\delta^{(k)}} \| r(c^{(k)}) + J_R(c^{(k)}) \delta^{(k)} \|_2^2, \tag{2.45}$$

o qual é equivalente a resolver as equações normais

$$J_R^T(c^{(n)}) J_R(c^{(n)}) \delta^{(n)} = -J_R^T(c) r(c). \tag{2.46}$$

Com isso, dada uma aproximação inicial $c^{(1)}$, a **iteração do método de Gauss-Newton** consiste em

$$J_R^T(c^{(k)}) J_R(c^{(k)}) \delta^{(k)} = -J_R^T(c) r(c) \tag{2.47}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{2.48}$$

Exemplo 2.2.2. A aplicação da iteração de Gauss-Newton ao problema de mínimos quadrados discutido no Exemplo 2.2.1 nos fornece a mesma solução obtida naquele exemplo (preservadas a aproximação inicial e a tolerância de precisão).
A implementação do método de Gauss-Newton para este problema no `GNU Octave` pode ser feita com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);
```

```
#residuo
r = @(c) y - f(x,c);

#jacobiana
function A = J(c)
  global x
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
endfunction

#aprox. inicial
c = [1.4 -1.8]';

#iteracoes de Gauss-Newton
k=0;
do
  k+=1;
  delta = - inv(J(c)'*J(c))*J(c)'*r(c);
  c = c + delta;
  [k,c',norm(delta)]
until ((k>10) | (norm(delta)<1e-4))
```

O método de Gauss-Newton pode ser lentamente convergente para problemas muito não lineares ou com resíduos grandes. Nesse caso, métodos de Gauss-Newton com amortecimento são alternativas robustas [1, 3]. Na sequência, introduziremos um destes métodos, conhecido como método de Levenberg-Marquardt.

2.2.2 Método de Levenberg-Marquardt

O método de Levenberg-Marquardt é uma variação do método de Gauss-Newton no qual a direção de busca $\delta^{(n)}$ é obtida da solução do seguinte problema regularizado

$$\min_{\delta^{(k)}}\{\|r(c^{(k)}) + J_R(c^{(k)})\delta^{(k)}\|_2^2 + \mu^{(k)}\|\delta^{(k)}\|_2^2\} \tag{2.49}$$

ou, equivalentemente,

$$\min_{\delta^{(k)}} \left\| \begin{bmatrix} r(c^{(k)}) \\ 0 \end{bmatrix} + \begin{bmatrix} J_R(c^{(k)}) \\ \mu^{(k)} I \end{bmatrix} \delta^{(k)} \right\|_2^2 \tag{2.50}$$

A taxa de convergência das iterações de Levenberg-Marquardt é sensível a escolha do parâmetro $\mu^{(k)} \geq 0$. Aqui, faremos esta escolha por tentativa e erro. O leitor pode aprofundar-se mais sobre esta questão na literatura especializada (veja, por exemplo, [1,3]).

Observação 2.2.1. Quando $\mu^{(k)} \equiv 0$ para todo n, o método de Levenberg-Marquardt é equivalente ao método de Gauss-Newton.

Exemplo 2.2.3. Consideremos o problema de mínimos quadrados discutido no Exemplo 2.2.1. O método de Gauss-Newton falha para este problema se escolhermos, por exemplo, $c^{(1)} = (0, 0)$. Isto ocorre pois, para esta escolha de $c^{(1)}$, a jacobiana $J(c^{(1)})$ não tem posto completo. Entretanto, o método de Levenberg-Marquardt com $\mu^{(k)} = 0{,}1$ é convergente, mesmo para esta escolha de $c^{(1)}$.
A implementação no `GNU Octave` do método de Levenberg-Marquardt (com $\mu^{(k)} = 0{,}1$ constante) para este problema pode ser feita com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);

#residuo
r = @(c) y - f(x,c);

#jacobiana
function A = JR(c)
  global x;
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
```

```
endfunction

#aprox. inicial
c = [0 0]';

#param. de amortecimento
mu = 0.1;

#iteracoes de Gauss-Newton
k=0;
do
  k+=1;
  JJ = [JR(c);mu*eye(2,2)];
  delta = - inv(JJ'*JJ)*JJ'*[r(c);zeros(2,1)];
  c = c + delta;
  printf("%d %1.1e %1.3e %1.3e\n", k,norm(delta),c')
until ((k>10) | (norm(delta)<1e-4))
```

Em construção ...

Licença CC-BY-SA 4.0

Resposta dos Exercícios

E 2.1.1. $c_1 = -1{,}3259$, $c_2 = 8{,}66071\text{E}{-2}$, $\|r(c)\|_2 = 1{,}01390$.

E 2.1.2. $c_1 = -4{,}50361\text{E}{-1}$, $c_2 = -2{,}78350\text{E}{-1}$, $c_3 = 1{,}46291$, $c_4 = 2{,}09648$, $\|r(c)\|_2 = 5{,}71346$

Referências Bibliográficas

[1] A. Björk. *Numerical methods for least squares problems*. SIAM, 1996.

[2] R.L. Burden, J.D. Faires, and A.M. Burden. *Análise Numérica*. CENGAGE Learning, 10. ed. edition, 2015.

[3] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 2006.

Índice Remissivo