Instituto Tecnológico de Aeronáutica – ITA
Matemática Computacional – CCI-22
Laboratório 7 – Ajuste de Curvas

Professor: Marcos Ricardo Omena de Albuquerque Maximo

2 de maio de 2019

1 Tarefas

1.1 Implementação

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado):

1. `c = MinimosQuadrados(Phi, x, y)`: realiza ajuste de curva usando o Método dos Mínimos Quadrados (MMQ). Os vetores coluna $\mathbf{x} = [x_1, x_2, \ldots, x_m]^T$ e $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ representam os m pontos $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ para os quais se deseja ajustar uma função $f^*(x)$ com modelo:

$$f^*(x) = c_0\Phi_0(x) + c_1\Phi_1(x) + \cdots + c_n\Phi_n(x) = \mathbf{c}^T\mathbf{\Phi}(x) \tag{1}$$

em que $\mathbf{\Phi}(x) = [\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)]^T$ e $\mathbf{c} = [c_0, c_1, \ldots, c_n]^T$. No MMQ, determina-se $\mathbf{c}$ de modo a minimizar a seguinte função de custo:

$$R(\mathbf{c}) = \sum_{i=1}^{m} \left(f^*(x_i) - y_i\right)^2 = \sum_{i=1}^{m} \left(\mathbf{c}^T\mathbf{\Phi}(x_i) - y_i\right)^2 \tag{2}$$

2. `[c, r] = RegressaoLinear(x, y)`: realiza Regressão Linear de modo a obter a melhor reta que aproxima os pontos m pontos $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ no sentido de mínimos quadrados. Assim, tem-se um caso particular de MMQ em que o modelo para $f^*(x)$ é:

$$f^*(x) = c_0 + c_1 x = \mathbf{c}^T \begin{bmatrix} 1 \\ x \end{bmatrix} \tag{3}$$

em que $\mathbf{c} = [c_0, c_1]^T$. Além disso, a função calcula o coeficiente de correlação r como estimativa da qualidade do ajuste:

$$r = \sqrt{\frac{R_M - R}{R_M}} \tag{4}$$

em que:

$$R = \sum_{i=1}^{m} \left(f^*(x_i) - y_i\right)^2 = \sum_{i=1}^{m} \left(c_0 + c_1 x_i - y_i\right)^2 \tag{5}$$

$$R_M = \sum_{i=1}^{m} \left[\left(\frac{\sum_{i=1}^{m} y_i}{m}\right) - y_i\right]^2 \tag{6}$$

3. `c = RegressaoLinear2D(x1, x2, y)`: realiza Regressão Linear com duas variáveis independentes x_1 e x_2. Os vetores coluna $\mathbf{x}_1 = [x_{11}, x_{12}, \ldots, x_{1m}]^T$, $\mathbf{x}_2 = [x_{21}, x_{22}, \ldots, x_{2m}]^T$ e $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ representam os m pontos $\{(x_{1i}, x_{2i}, y_i)\}$, $i = 1, \ldots, m$ para os quais se deseja ajustar uma função de duas variáveis $f^*(x_1, x_2)$ que define um plano no $\mathbb{R}^3$:

$$f^*(x_1, x_2) = c_0 + c_1 x_1 + c_2 x_2 \tag{7}$$

Assim, a função determina $\mathbf{c} = [c_0, c_1, c_2]^T$ que minimiza a seguinte função de custo:

$$R(c_0, c_1) = \sum_{i=1}^{m} \left(f^*(x_i) - y_i\right)^2 = \sum_{i=1}^{m} \left(c_0 + c_1 x_{1i} + c_2 x_{2i} - y_i\right)^2 \tag{8}$$

1.2 Análise

1. O aluno Saraiva estava na sala da ITAndroids testando o sistema de arremesso que projetou para seu robô de LEGO arremessar bolinhas durante partidas de THBall. Devido a um erro de projeto, as bolinhas acabaram sendo arremessadas muito para o alto e assim Saraiva percebeu que elas quicavam várias vezes no chão antes de parar. Por ser um aluno muito curioso, Saraiva se perguntou qual seria o coeficiente de restituição do choque entre bolinha e chão para que ela quicasse tanto. Observando a trajetória da bolinha, ele percebeu que a cada novo choque, a bolinha passava menos tempo no ar até o próximo choque. Considere t_i o instante do choque i, o tempo que a bolinha passa no ar entre o choque i e $i+1$ é $T_i = t_{i+1} - t_i$. Desprezando a resistência do ar, o tempo para que a bolinha chegue no ponto mais alto da trajetória no intervalo $[t_i, t_{i+1}]$ vale $\frac{T_i}{2}$. Neste ponto, a velocidade é nula. Portanto, tem-se:

$$v(t) = v_i - gt \Rightarrow v(\frac{T_i}{2}) = 0 = v_i - g\frac{T_i}{2} \Rightarrow T_i = 2\frac{v_i}{g} \tag{9}$$

em que t é o tempo medido a partir de t_i, $v(t)$ é a velocidade vertical da bolinha no instante t, v_i é a veloidade vertical da bolinha logo após o choque i e g é a aceleração da gravidade. Além disso, a velocidade vertical v_{i+1}^- logo antes do choque $i+1$ é dada por:

$$v(t) = v_i - gt \Rightarrow v(T_i) = v_{i+1}^- = v_i - gT_i = v_i - 2g\frac{v_i}{g} \Rightarrow v_{i+1}^- = -v_i \tag{10}$$

Seja ε o coeficiente de restituição do choque, tem-se:

$$v_{i+1} = -\varepsilon v_{i+1}^{-} \Rightarrow v_{i+1} = \varepsilon v_i \tag{11}$$

Substituindo (9) em (11), tem-se:

$$T_{i+1} = \varepsilon T_i \tag{12}$$

Ademais, considerando que a bolinha é lançada a partir de um ponto muito próximo ao chão, pode-se considerar também:

$$T_0 = 2\frac{v_0}{g} \tag{13}$$

em que v_0 é a velocidade vertical inicial da bolinha. Desse modo, a partir de (12), podemos obter o seguinte modelo para os tempos T_i:

$$T_i = \varepsilon^i T_0 \Rightarrow T_i = T(i) = \varepsilon^i \left(\frac{2v_0}{g}\right) \tag{14}$$

Após rapidamente perceber as relações acima, Saraiva decidiu filmar um lançamento de bolinha e tentar determinar os tempos T_i a partir do vídeo gerado. Porém, devido à dificuldade de identificar precisamente os tempos de choque, ele obtém para cada T_i um T_i' medido tal que:

$$T_i' = T_i + \xi \tag{15}$$

em que ξ é um número aleatório distribuído uniformemente no intervalo $[-s, s]$. Observe que com os T_i' obtidos, é possível realizar uma regressão linear para obter ε e v_0 usando o modelo de (14), após uma devida troca de variáveis. O arquivo `bolinha.mat` contém vetores $\mathbf{T} = [T_1, \ldots, T_N]^T$ e $\mathbf{T'} = [T_1', T_2', \ldots T_N']^T$ para uma dada realização do experimento. Determine ε e v_0 para este caso. Utilize $g = 9,8m/s^2$. Compare em um gráfico a curva verdadeira dada por $\mathbf{T}$ e a ajustada pela regressão a partir das medidas corrompidas por ruído.

A função `[T, Tp] = SimularBola(epsilon, v0, N, s)` simula o experimento descrito para $i = 1 : N$. Experimente simular o experimento para alguns valores de s e verifique como varia o coeficiente de correlação r. Para padronizar, use $\varepsilon = 0,95$, $v_0 = 2m/s$, $N = 5$ e $s \in \{0,01 : 0,01 : 0,1\}$. Tire uma média de 1000 realizações do experimento para cada valor de r para reduzir o ruído. Mostre o resultado (variação de r médio com variação de s) em um gráfico. Este resultado é esperado?

2. Deseja-se determinar um modelo para o tempo computacional da função de programação `s = FuncaoCaixaPreta(n1, n2)` a partir dos valores de seus parâmetros sem necessitar analisar sua implementação, i.e. com testes do tipo caixa preta (na prática, frequentemente não temos acesso à implementação de uma função). Após alguns testes, concluiu-se que o modelo de tempo computacional deve ser:

$$f_T(n_1, n_2) = \gamma n_1^{\alpha} n_2^{\beta} \tag{16}$$

em que α, β e γ são parâmetros. A função `T = MedirTempoFuncaoCaixaPreta(x1, x2)` realiza medições de tempo computacional da função `s = FuncaoCaixaPreta(n1, n2)`. Observe que $\mathbf{x}_1 = [x_{11}, x_{12}, \ldots, x_{1m}]^T$ e $\mathbf{x}_2 = [x_{21}, x_{22}, \ldots, x_{2m}]^T$ definem os m casos de teste $(n_1, n_2) \in \{(x_{1i}, x_{2i})\}, i = 1, \ldots, m$. Além disso, o retorno da função é $\mathbf{T} = [T_1, T_2, \ldots, T_m]^T$ tal que $T(i) = f_T(x_{1i}, x_{2i}), i = 1, \ldots, m$. Na prática, faz-se a média de $N_t = 30$ medidas de tempo para cada caso para redução de ruído.

Assim, meça tempos computacionais no conjunto de casos de teste $(n_1, n_2) \in \{10 : 10 : 100\} \times \{10 : 10 : 100\}$ (note que neste caso tem-se $m = 10 \times 10 = 100$ casos de teste) e utilize Regressão Linear Múltipla para ajustar os parâmetros α, β e γ do modelo $f_T(n_1, n_2)$. Exiba em um gráfico os m pontos usados na regressão e a superfície definida por $f_T(n_1, n_2)$. Finalmente, responda:

- Qual a complexidade computacional do algoritmo de `FuncaoCaixaPreta` na notação $\mathcal{O}$?
- Finalmente verifique o código da `FuncaoCaixaPreta`. A complexidade obtida experimentalmente condiz com a esperada por análise do algoritmo?

2 Instruções

- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos .m com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos .m fornecidos para evitar erros.

- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida quanto à permissão de uso de alguma função ou comando, recomenda-se consultar o professor.

- Não é necessário reimplementar métodos de laboratórios anteriores. Assim, funções implementadas em laboratórios anteriores podem ser utilizadas. Caso prefira, também é permitido utilizar funções equivalentes do MATLAB.

- Não é necessário se preocupar com verificação dos dados de entrada: assuma que x e y tem mesma dimensão etc.

- Os arquivos .m implementados devem ser entregues juntamente com um relatório.

- No relatório, não é necessário demonstrar que as funções implementadas funcionam corretamente (isto será verificado separadamente). Basta incluir resultados e conclusões relativos à **Análise**.

- Para facilitar a correção da Análise, inclua os gráficos diretamente no relatório. Nos gráficos, coloque títulos, legendas e nomes nos eixos.

3 Dicas:

- Para carregar o arquivo `bolinha.mat`, faça: `load 'bolinha.mat'`.
- A função logaritmo nepeariano no MATLAB é `log`.
- Não precisa usar a função `MetodoMinimosQuadrados` para implementar as demais, dado que já há fórmulas deduzidas para Regressão Linear nos slides.
- Utilize `cell` para definir o vetor de funções `Phi`: `Phi{1} = @(x) x`, `Phi{2} = @(x) x^2`, `Phi{3} = @(x) cos(x)` etc.
- Para criar os m pontos necessários no item 2 da Análise, use:

```
[X1, Y1] = meshgrid(10:10:100, 10:10:100);
m = size(X1, 1) * size(X1, 2);
x1 = reshape(X1, m, 1);
x2 = reshape(X2, m, 1);
```

- Para plotar gráficos 3D, use:
  - `surf(X1, X2, Y)`: plota uma superfície 3D.
  - `plot3(x1, x2, y, '*')`: plota um conjunto de pontos em 3D.
- Cuidado com o formato requerido para vetores para evitar problemas na correção automática.
- Para criar gráficos com alta qualidade em formato PNG para inclusão em arquivos do Microsoft Word, utilize o comando: `print -dpng -r300 grafico.png`.
- Se utilizar LaTeX, dê preferência para incluir gráficos em formato vetorizado. No Linux, utilizando `pdflatex`, você pode gerar um gráfico em formato EPS usando "`print -depsc2 grafico.eps`" e depois convertê-lo para PDF usando o comando de terminal "`epstopdf grafico.eps`". O arquivo PDF é aceito pelo `pdflatex`.