Relatório - miniprojeto 4

Laís Saloum Deghaide, nUSP: 11369767

Parte 1

1. Introdução

Neste projeto o cliente nos pediu para encontrar a máxima diferença entre o valor mínimo e o valor máximo encontrados para a espessura da chapa metálica medida em 7 pontos.

Para isso, utilizaremos a interpolação. Ela é um método eficiente para encontrarmos o polinômio que descreve a chapa que queremos é frequentemente usada em modelagem e análise numérica.

Sendo assim, construiremos um polinômio de grau 6, sabendo que a espessura h_1 a h_7 de uma chapa metálica são medidas em 7 posições dadas pelo vetor
$x = (x_1 = 0,\ x_2 = 0,1,\ x_3 = 0.2,\ x_4 = 0.5,\ x_5 = 0.8,\ x_6 = 0.9,\ x_7 = 0.1).$

Objetivo

Temos que h(x) pode ser aproximado por um polinômio de grau $\leq$ 6. Como meu número USP acaba em 7, calculei a máxima diferença A, dada por:

$$A = max_x h(x) - min_x h(x)$$

E o erro, dado por:

$$erro = \frac{|A_2 - A_1|}{max(|A2|, max|A1|)} \leq 1\%,$$

sendo que $max(|A2|,\ max|A1|)$ é o maior valor entre o módulo de A_1 e A_2.

2. Desenvolvimento

Para isso, precisei encontrar A_1 e A_2, sendo A_1 a diferença de $max_x h(x) - min_x h(x)$ calculada numa amostra de 40 pontos, e A_2 a diferença de $max_x q(x) - min_x q(x)$ calculada numa amostra de 80 pontos. Sendo a máxima diferença mais exata em A_2, já que é uma amostra maior.

Temos que h(x) é a interpolada que relaciona os valores do vetor x com as espessuras da chapa metálica e é a que devemos encontrar.

Para tal, foi utilizada a interpolação polinomial de Lagrange, dada por:

$L = \{L_i\ f = f(x_i)\}$ com os pontos $x_1, \dots, x_n$ distintos (reais ou complexos), e $V = P_{n-1}$.

O cálculo da interpolada foi feito da seguinte forma:

```
display("Miniprojeto 4 parte 1")

function res = resultado(h)
  x = [0; 0.1; 0.2; 0.5; 0.8; 0.9; 1];

  % Valores que estarão no polinomio p para encontrar a máxima diferença
  % c1 possui maior intervalo entre os pontos (0.001), e é minha primeira estimativa, menos aproximada, para encontrar o A
  c1 = 0:0.001:1;
  c2 = 0:0.0005:1;


  % Criando a matriz M*a = h, a fim de ter a para montar meu polinômio
  n=length(x);
  M=ones(n,1);

  for k = 1:(n-1)
   M = [M x.^k];
  endfor
  a = M\h;
```

Obs: c_1 e c_2 são os vetores, cujos valores são gerados entre 0 e 1, necessários para que eu possa encontrar A_1 e A_2

Obs 2: a matriz M é a nossa base 1, x, x^2...

Obs 3: **a** representa os coeficientes do polinômio

Obs 4: o grau do polinômio se ajusta com a quantidade de pontos.

Tendo os coeficientes **a**, passamos para a construção, de fato, do nosso polinômio desejado h(k):

```
% Construindo o polinômio p e por consequencia achando A1
for k=1:length(c1)
  p(k) = a(1);

  for m = 1:(n-1)
    p(k) = p(k) + a(m+1) * c1(k) ^ m;
  endfor
endfor
A1 = max(p) - min(p);

% Construindo o polinomio q e por consequencia achando A2, diferença mais aproximada
for k=1:length(c2)
  q(k) = a(1);

  for m = 1:(n-1)
    q(k) = q(k) + a(m+1) * c2(k) ^ m;
  endfor
endfor
A2 = max(q) - min(q);
```

Obs: como é necessário ter dois cálculos de máxima diferença do polinômio para calcular o erro, então, é necessário também ter dois polinômios, q e p, calculados no código acima

Depois de calcular cada um dos polinômios necessários, encontrei as máximas diferenças A_1 e A_2 (cálculo exposto na imagem acima).

E tendo A_1 e A_2 bastou calcular o erro e verificar se ele era menor que 1%.

```
% Cálculo do erro
maxi = max(abs(A2), abs(A1))

if(maxi == 0)
  erro = 0
else
  erro = (abs(A2-A1)) / maxi
endif

% Checando se o erro é menor que 1%
if(erro <= 0.01)
  res = A2;
else
  res = -1;
endif
```

Obs: como o erro é calculado por uma divisão, o divisor não pode ser zero, e portanto, faço essa verificação.

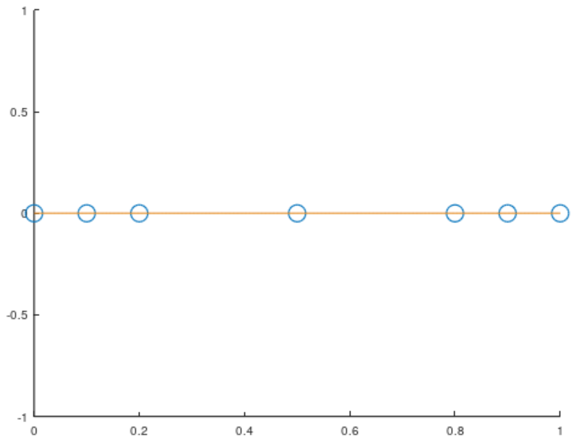
3. Análise gráfica e Resultados

Primeiro, é importante checarmos se a interpolada está bem definida no caso em que todos os valores do vetor h são zero. Neste caso, o teremos o erro igual a zero e $A = max_x h(x) - min_x h(x)$ também sendo zero. O gráfico então deve ser uma reta.

Para termo a análise gráfica, o código ficou assim:

```
% Gráfico da interpolada
scatter(x, h)
hold on
plot(c1, p, c2, q)
```

```
>> resultado([0;0;0;0;0;0;0])
maxi = 0
erro = 0
ans = 0
```

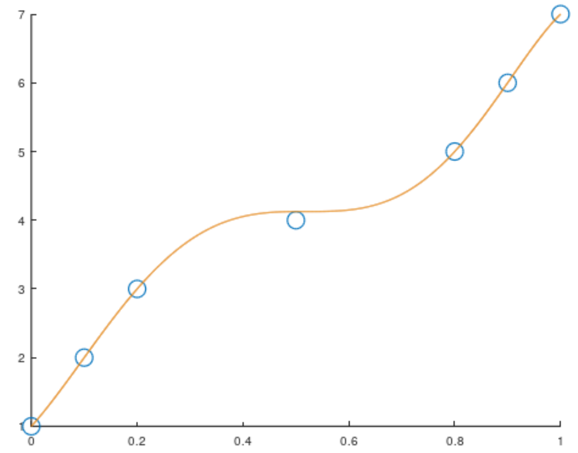


Obtivemos o resultado que esperávamos.

Realizando a mesma análise, só que agora com valores diferentes em h:

```
>> resultado([1;2;3;4;5;6;7])
maxi = 6.0000
erro = 0
ans = 6.0000
```

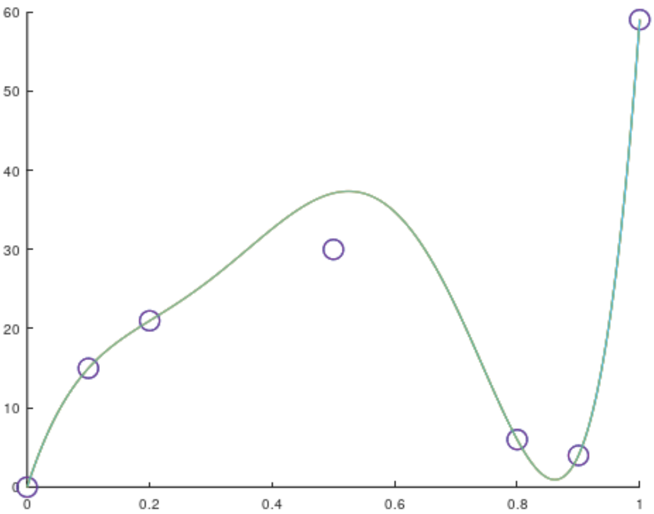
Encontramos que o valor máximo é 6.00 e é equivalente a A_2.
Visualizando graficamente, temos:



Analisando o resultado da interpolada nos seguintes pontos:

```
>> resultado([0;15;21;30;6;4;59])
maxi = 59.000
ans = 59.000
```

Note que o polinômio encontrado foi bem assertivo:



Podemos concluir que a interpolação é um método muito eficiente para modelagem e análise numérica. E neste trabalho, foi possível verificar que o problema algébrico da interpolação tem solução para quaisquer h_1, ... , h_m, visto que o sistema é linearmente independente, possibilitando encontrarmos o melhor polinômio para o cálculo da chapa metálica.

Parte 2

1. Introdução

Nessa segunda parte, os dados são fornecidos em dois vetores: x (contendo as posições entre 0 e 1) e h (contendo os valores medidos). Além disso, as espessuras (h_n) da chapa metálica foram medidas não somente em 7 posições como na parte 1, mas em um número bem maior em x_n.

Sabe-se que $h(x)$ pode ser bem aproximada por um polinômio de grau ≤ 6 e que o ruídos das medições é Gaussiano não correlacionado de variância constante. Portanto, nesta parte queremos que o erro quadrático médio seja minimizado e também queremos encontrar a máxima diferença de h.

2. Objetivos

Devemos criar uma função f(x) do espaço V que maximiza a verossimilhança dos dados, ou seja, aquela que minimiza o erro quadrático médio entre $f(x_i)$ e y_i, sendo y_i observações independentes, tal que:

$$y_i = f(x_i) + \epsilon_i$$

3. Desenvolvimento

Para isso, criei as seguintes funções auxiliares:

```
%Função que gera as bases do polinômio.
function res = phi(k, x)
  res=x.^(k-1);
end

%Função que gera a função f
function res = fmodel(theta, x)
  m = length(theta);
  res = theta(1)*phi(1,x);
  for k = 2:m
   res = res+theta(k)*phi(k, x);
  end
end

%Função que gera a função perda
function ll = perda(theta)
  global xx %% vetor linha
  global yy %% vetor linha
  x = xx;
  y = yy;
  nx = length(x);
  fx = fmodel(theta,x);
  ll = (fx-y)*(fx-y)'/nx;
end

%Função que calcula a derivada parcial da função f
function res = dfmodel(theta,x,k)
  res = phi(k, x);
end

%Função que calcula o gradiente da função perda
function res = gradperda(theta)
  global xx
  global yy
  x = xx;
  y = yy;
  nx = length(x);
  m = length(theta);
  fx = fmodel(theta, x);
  for k = 1:m
    dfx = dfmodel(theta ,x, k);
    res(k)  = 2/nx*(y-fx)*(-dfx');
  end
end
```

- A função phi(k, x) é responsável por gerar as bases do polinômio;
- A função fmodel é responsável por gerar f, estimador de máxima verossimilhança, de acordo com o vetor com valores de θ os valores das abscissas e as bases do polinômio;
- A função perda é responsável por calcular a perda a partir do erro quadrático médio entre $f(x_i)$e $f(h_i)$;
- A função dfmodel é responsável por calcular a derivada parcial da função f em relação a θ, que resulta nas bases da função phi(k,x);

- A função gradperda é responsável por calcular o gradiente da função perda usada na regra de avanço para achar o próximo valor de theta.

E na função principal recebi x e h como parâmetros, gerei um θ aleatório para iniciar o nosso cálculo responsável por achar o polinômio da chapa desejável.

```
function res=resultado2(x, h)
  global xx
  global yy

  % tornando meus paramentros globais
  xx = x;
  yy = h;

  m = 7;

  %% condicao inicial randomica N(0,1)
  theta = randn(1, m);

  %% parametros
  lrate = 0.235;
  iter = 1;
  tol = 0.001;
  update = 1e10;
  itermax = 5000;
```

Os parâmetros $lrate, iter, tol, update, itermax$ são os parâmetros necessários para calcular o método do gradiente, dado por um θ inicial arbitrário menos a multiplicação do learning rate (lrate) com o gradperda,

Exemplo: Método do gradiente, ou de máxima descida

$$\theta^0 \text{ arbitrário }, \qquad \theta^{k+1} = \theta^k - \alpha \nabla \mathcal{L}(\theta^k), \qquad \text{if } \|\theta^{k+1} - \theta^k\| < \text{ tol , then STOP}$$

Fórmula retirado dos slides do professor

Realizando os cálculos das iterações responsáveis por encontrar o θ que melhor se encaixa na função e no final das iterações obter o polinômio da chapa que queremos:

```
%% iteracoes
while (update > tol && iter < itermax)
  resperda(iter) = perda(theta);
  dtheta = -lrate*gradperda(theta);
  update = norm(dtheta);
  iter = iter+1;
  theta = theta+dtheta;
  postagem = [iter-1 resperda(iter-1) update]
```

Como queremos encontrar a função que melhor se ajusta aos pontos passados pelo parâmetro, foi necessário realizar uma série de teste modificando os valores do $lrate$ e $itermax$ a fim de ajustar e encontrar o melhor θpara o polinômio desejável.

4. Análise gráfica e Resultados

Como resultado dos testes realizados na fase de desenvolvimento, encontrei que o melhor valor para o learning rate e número máximo de iterações foram: 0,235 e 5000 respectivamente.

Então, vamos analisar o resultado de $A = \max_x h(x) - \min_x h(x)$ sendo $x = [0\ 0.1\ 0.2\ 0.5\ 0.8\ 0.9\ 1]$ e $h = [1\ 2\ 3\ 4\ 5\ 6\ 7]$, mesmos valores que foi realizado na parte 1 do projeto, sendo o learning rate = 0,235 e máximo de iterações = 5000. Como já sabemos, o resultado deve ser 6.00.

```
theta
c = 0:0.0005:1;
h = fmodel(theta, c);
A = max(h)-min(h);
res = A;
```

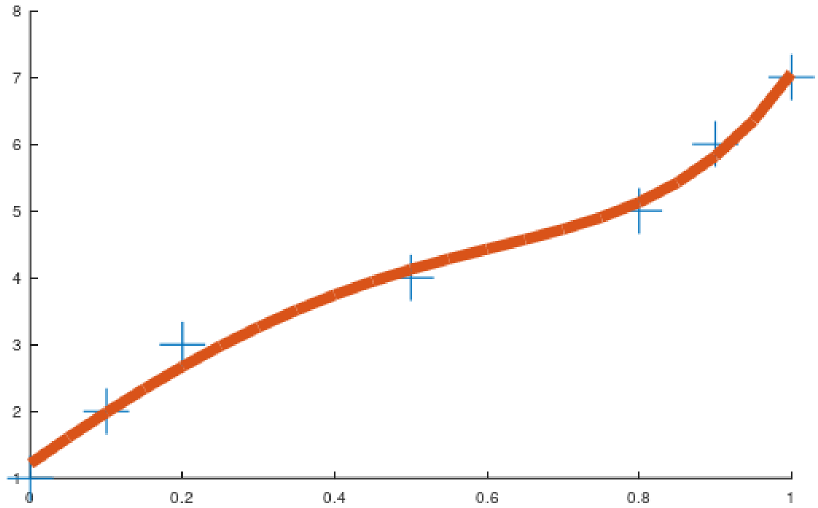
Parte do código responsável por encontrar o h e calcular o A.

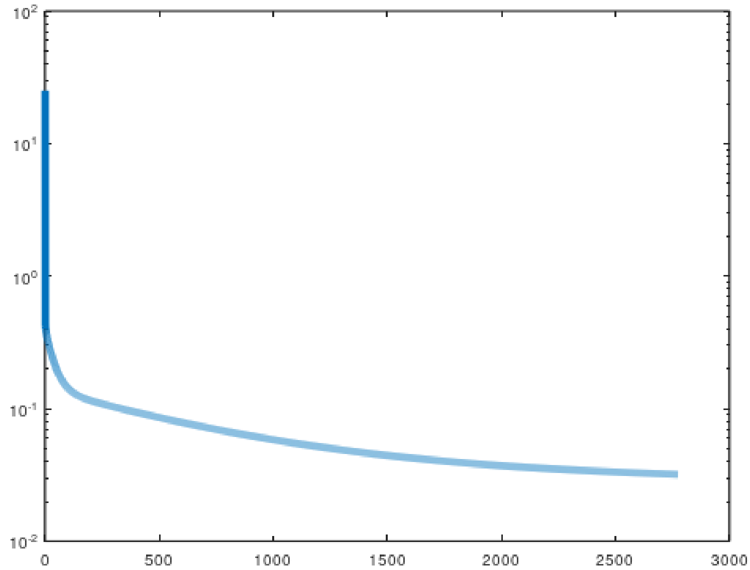
O resultado foi o seguinte:

```
postagem =

   2.7740e+03   3.1966e-02   9.9980e-04

Resultado do metodo do gradiente:
theta =

   1.2176   7.9870  -2.9125  -3.3093  -1.0164   2.4480   2.6441

ans = 5.8409
```

Podemos perceber que o resultado foi bem próximo de 6.00. Podemos concluir que de fato, realizando os cálculos pelo método gradiente também conseguimos encontrar um resultado coerente para o nosso polinômio. Possivelmente, se eu colocasse um número maior de itermax e ajustasse o learning rate, chegaria a exatamente 6.00, mas por questões de desempenho da minha máquina, com 5000 itermax já está devagar para encontrar o resultado.

Visualizando graficamente:





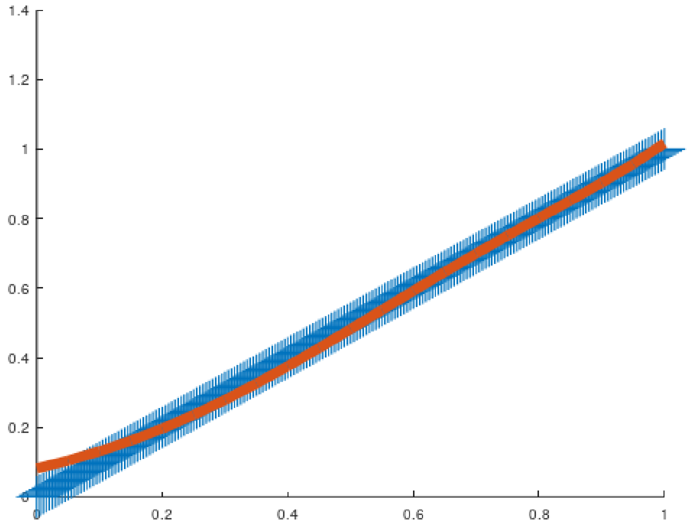
Como nesta segunda parte não temos apenas 7 pontos, também fiz mais algumas análises com uma quantidade maior de pontos:

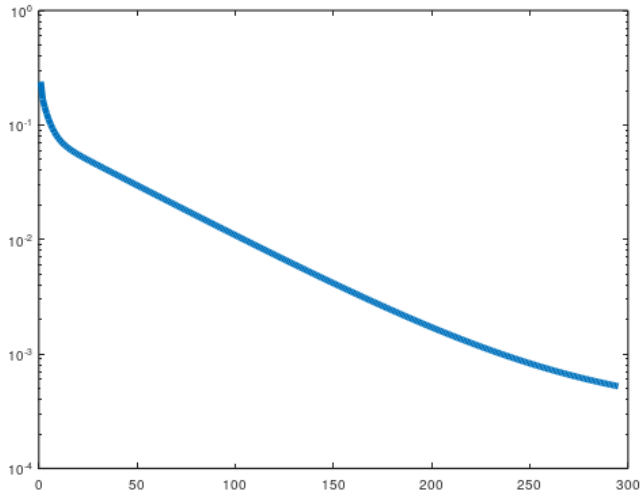
Tentei aproximar a função para uma função afim x=h:

```
resultado2(0:0.005:1, 0:0.005:1)

Resultado do metodo do gradiente:
theta =

   0.082071    0.416941    0.669713    0.720929   -0.727416   -1.149972    1.004207

ans = 0.9344
```

Testei com 200 pontos entre 0 e 1 e obtive como resposta 0.93, bem aproximado de 1 (valor que ideal)

Visualizando graficamente:

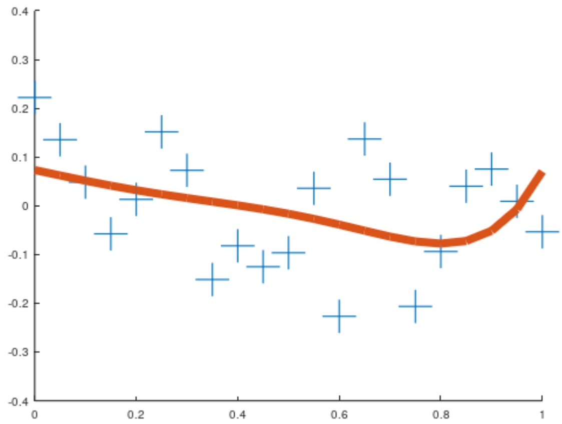


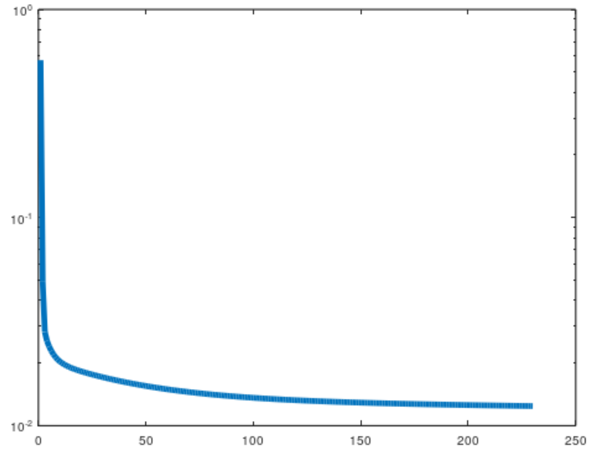


Realizando com 21 pontos:

```
>> resultado2(0:0.05:1, 0.1*randn(1,21))

Resultado do metodo do gradiente:
theta =

   0.073236  -0.221056  -0.049734   0.794696  -0.644015  -1.770634   1.887988

ans = 0.1509
```

Visualizando graficamente:





Realizando com 201 pontos:

```
Miniprojeto 4 parte 2
>> resultado2(0:0.005:1, 0.1*randn(1,201))

postagem =

   3.5000e+01   1.0569e-02   9.5708e-04

Resultado do metodo do gradiente:
theta =

   0.081607  -0.323708  -0.401862   1.180147   0.184716   0.125365  -0.931482

ans = 0.1668
```

Visualizando graficamente: