PONTIFÍCIA UNIVERSIDADE CATÓLICA GOIÁS
CURSO DE ENGENHARIA DE COMPUTAÇÃO
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO

AED IV – *SPLINES* CÚBICOS

GOIÂNIA 2018

PONTIFÍCIA UNIVERSIDADE CATÓLICA GOIÁS

AED IV – *SPLINES* CÚBICOS

LUCAS MACEDO DA SILVA

GOIÂNIA, 2018

PONTIFÍCIA UNIVERSIDADE CATÓLICA GOIÁS

AED IV – *SPLINES* CÚBICOS

Trabalho apresentado por Lucas Macedo da Silva para avaliação e aplicação dos conceitos aprendidos em atividade extra disciplinar.

GOIÂNIA, 2018

Sumário

1 INTRODUÇÃO

A interpolação de n+1 pontos de polinômios de grau n, implica em erros de arredondamento e de estimativa (CHAPRA; CANALE, 2011). Os splines tem o intuito de interpolar esses pontos a partir de polinômios de grau menor. Quando ocorre por curvas de terceiro grau são conhecidos como *splines* cúbicos.

Os *splines* se sobressaem quando comparados com outros métodos de interpolação, visto que possuem um grau menor, bem como apresenta menor número de oscilações. Fornecendo, portanto, uma melhor aproximação do comportamento de funções com grandes variações (CHAPRA; CANALE, 2011).

Um *spline* natural é aquele em que a segunda derivada aplicada nos pontos iniciais e finais são iguais. Isso implica na diminuição de variáveis a serem calculadas e garante que o *spline* passará pelos pontos extremos. Já os *splines* extrapolados, também conhecidos como *splines not-a-knot*, são *splines* em que as terceiras derivadas aplicadas em dois pontos sucessivos, são iguais (JUSTO et al., 2018).

2 *SPLINES*

Um *spline* permite interpolar pontos de um dado problema. Sendo um conjunto de n pontos então sua equação é definida por:

$si(x) = ai(x - xi)^3 + bi(x - xi)^2 + ci(x - xi) + di$, com i = 0, 1, ..., n-1

A equação deve satisfazer as seguintes condições.

I. $si(xi) = yi$ e $sn-1(xn) = yn$
II. $si(xi+1) = si+1(xi+1)$
III. As concavidades e inclinações devem ser mantidas, ou seja, $s'i(xi+1) = s'\, i+1(xi+1)$ e $s''i\,(xi+1) = s''\, i+1(xi+1)$

Se igualarmos x á xi, obtemos que s(xi) = di.

$$hi = xi+1 - xi$$

O coeficiente a pode ser definido por:

$$ai = \frac{s''i + 1(xi + 1) - s''i(xi)}{6hi}$$

Já o coeficiente b pode ser definido por:

$$bi = \frac{s''i(xi)}{2}$$

Sendo dyi, definido por:

$$dyi = \frac{yi + 1 - yi}{hi}$$

Pode-se definir o coeficiente c, a partir de:

$$ci = dyi - \frac{s''i + 1(xi + 1)\ +\ si(xi)}{hi} hi$$

Ao final obtém-se um sistema na forma:

Figura 2 – Sistema Linear *Spline* Natural

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & \ddots & \ddots & \ddots \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \begin{bmatrix} s_1''(x_1) \\ s_2''(x_2) \\ s_3''(x_3) \\ \vdots \\ s_{n-1}''(x_{n-1}) \end{bmatrix} = 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix}$$

Para … rem iguais tem-se o seguinte sistema linear:

Figura 3 – Sistema Linear *Spline* extrapolado

$$\begin{bmatrix} \frac{(h_0+h_1)(h_0+2h_1)}{h_1} & \frac{h_1^2-h_0^2}{h_1} & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \frac{h_{n-2}^2-h_{n-1}^2}{h_{n-2}} & \frac{(h_{n-2}+h_{n-2})((h_{n-1}+h_{n-2})}{h_{n-2}} \end{bmatrix}$$

$$\begin{bmatrix} s_1''(x_1) \\ s_2''(x_2) \\ \vdots \\ s_{n-2}''(x_{n-2}) \\ s_{n-1}''(x_{n-1}) \end{bmatrix} 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix}$$

4 APLICAÇÃO DOS *SPLINES* PARA A RESOLUÇÃO DO PROBLEMA PROPOSTO

Os *splines* são de deveras importância para o desenho de gráficos, visto que limitam o grau da função. Tornando-o menor que o grau obtido nos métodos tradicionais, onde com n+1 pontos obtém-se uma função de grau n.

4.1 Enunciado do problema

Figura 3 – Enunciado do problema

Seja a função

$$f(x) = \begin{cases} e^x & -2 \le x \le 0, \\ x\sin(5x)+1, & 0 \le x \le 4 \end{cases}$$

Definida por n pontos distintos $(-2, f(-2))$, $(-2+h, f(-2+h))$, $(-2+2h, f(-2+2h)), \dots, (4, f(4))$, com $h = (4-(-2))/(n-1)$

Construa a curva de $y = f(x)$ usando *splines* cúbicos, com $m = 121$ número de pontos a interpolar e $n = 7$ e $n = 13$

Faça o gráfico para os *splines* cúbicos natuarais e extrapolados

4.2 Código *Octave*

Para a resolução do problema foi desenvolvido um código no *Octave* que permita, calcular os *splines* e plotar os devidos gráficos.

```
function [a] = calc_a (s = [], h = [])
  n = length(h);
  for i = 1 : n
    if (i-1 == 0)
      a(i) =  (s(i,1)) / (6*h(i));
    else
      if (i == n)
        a(i) = (-s(i-1,1)) / (6*h(i));
      else
        a(i) = (s(i,1) - s(i-1,1)) / (6*h(i));
      endif
    endif
  endfor
endfunction
function [b] = calc_b(s = [])
  n = length(s) + 1;
  for  i = 1 : n
    if (i - 1 == 0)
      b(i) = 0;
    else
      b(i) = s(i-1)/2;
    endif
  endfor
endfunction
function [c] = calc_c(s = [], dy = [], h = [])
  n = length(s) + 1;
```

```
  for i = 1: n
    if (i-1 == 0)
      c(i) = dy(i) - (s(i,1)/6)*h(i);
    else
      if (i == n)
         c(i) = dy(i) - (2*s(i-1,1)/6)*h(i);
      else
        aux = s(i,1) + 2*(s(i-1,1));
        aux = aux / 6.0;
        c(i) = dy(i) - aux*h(i);
      endif
    endif
  endfor
endfunction
function [dy] = calc_dy(y = [], h = [])
  n = length(y) -  1;
  for i = 1 : n
    dy(i) = (y(i+1) - y(i)) / h(i);
  endfor
endfunction
function [h] = calc_h (x = [])
  n = length (x) - 1;
  for i = 1 : n
    h(i) = x(i+1) - x(i);
   endfor
endfunction
function [x, y] = calc_x_y(n)
  h = 6/(n-1);
  r = 0;
  i = 1;
```

```
  while r < 4
    r = -2 + i * h;
    x(i) = r;
    if r<=-2 || r <=0
      y(i) = f1(r);
    else
      y(i) = f2(r);
    endif
    i = i + 1;
  endwhile
endfunction
function [s] = calc_s(h = [], dy = [])
  %% m == s
  n = length(h) - 1;
  m = zeros(n);
  j = 2;
  %%Sistema: m*S'' = b
  %%Calculando a matriz m;
  for i = 1: n
    if (i-1 == 0);
      m(i, i) = 2 * (h(j-1) + h (j));
      m(i, i+1) = h(j);
    else
      if (i + 1 > n)
        m(i, i-1) = h(j-1);
        m(i, i) = 2 * (h(j-1) + h (j));
      else
        m(i, i) = 2 * (h(j-1) + h (j));
        m(i, i+1) = h(j);
        m(i, i-1) = h(j-1);
```

```
      endif
    endif
    j = j + 1;
  endfor
  %%Calculando o vetor de constantes
  n = length(dy) - 1;
  b = [];
  j = 2;
  for i = 1: n
    b(i,1) = 6*(dy(j) - dy(j-1));
    j = j + 1;
  endfor
  %%Resolvendo o sistema
  s = inv(m) * b;
endfunction
function [s] = calc_s(h = [], dy = [])
  %% m == s
  n = length(h) - 1;
  m = zeros(n)
  j = 2;
  %%Sistema: m*S'' = b
  %%Calculando m00 e mnn
  m(1,1) = (((h(1) + h(2))*(h(1) + 2*h(2))))/h(1);
  m(1,2) = ((h(1)^2 - h(2)^2))/h(1);
  m(n-1, n) = ((h(n-1)^2 - h(n)^2))/h(n-1);
  m(n, n) = (((h(n-1) + h(n-1)))*(h(n) + h (n-1)))/h(n-1);
  %%Calculando a matriz m;
  for i = 2: n - 1
    if (i + 1 > n)
      m(i, i-1) = h(j-1);
```

```
      m(i, i) = 2 * (h(j-1) + h (j));
    else
      m(i, i) = 2 * (h(j-1) + h (j));
      m(i, i+1) = h(j);
      m(i, i-1) = h(j-1);
    endif
    j = j + 1;
  endfor
  %%Calculando o vetor de constantes
  n = length(dy) - 1;
  b = [];
  j = 2;
  for i = 1: n
    b(i,1) = 6*(dy(j) - dy(j-1));
    j = j + 1;
  endfor
  %%Resolvendo o sistema
  s = inv(m) * b;
endfunction
function y = f1(x)
  y = e.^x;
endfunction
function y = f2(x)
  y = x * sin (5*x) + 1;
endfunction
function grafico (n)
  %% -------------- Natural ---------------------------
  x = [];
  y = [];
  [x, y] = calc_x_y(n);
```

```
%% n = length(x);
h = [];
dy = [];
a = [];
b = [];
c = [];
h = calc_h(x);
dy = calc_dy(y, h);
s = calc_s(h, dy);
a = calc_a(s, h);
b = calc_b(s)
c = calc_c(s, dy, h);
%% ------------------- Extrapolado --------------
x_e = [];
y_e = [];
[x_e, y_e] = calc_x_y(n);
n_e = length(x_e);
s_e = calc_s_extrapolado(h, dy);
a_e = calc_a(s_e, h);
b_e = calc_b(s_e);
c_e = calc_c(s_e, dy, h);
%%Gráficos
for i = 1:n-2
  ex = x(i):0.1:x(i+1);
  ey = a(i)*(ex - x(i)).^3 + b(i)*(ex - x(i)).^2 + c(i)*(ex - x(i)) + y(i);
  ex_e = x_e(i):0.1:x_e(i+1);
  ey_e = a_e(i)*(ex_e - x_e(i)).^3 + b_e(i)*(ex_e - x_e(i)).^2 + c_e(i)*(ex_e - x_e(i)) + y_e(i);
  hold on
  plot (ex, ey, 'b', ex_e, ey_e, 'r');
```

```
  endfor
  for i = 1:n-1
    hold on

    plot (x(i), y(i),'*', 'MarkerEdgeColor', 'k', 'LineWidth',3);

  endfor

endfunction
```

4.3 Gráficos obtidos

Ao final da execução do código pode-se verificar os gráficos obtidos, conforme figuras 4 e 5. Obtendo-se tanto os gráficos dos *splines* naturais quanto os gráficos dos *splines* extrapolados. Vale ressaltar que o gráfico dos *splines* naturais é aquele que possui coloração azul, linha constantes, já o gráfico dos *splines* extrapolados é aquele que possuem coloração vermelha, linhas tracejadas, os pontos se encontram com marcações em cor preta.

Gráfico para n = 7

Figura 4 – Gráfico do *spline* natural e extrapolado para n = 7

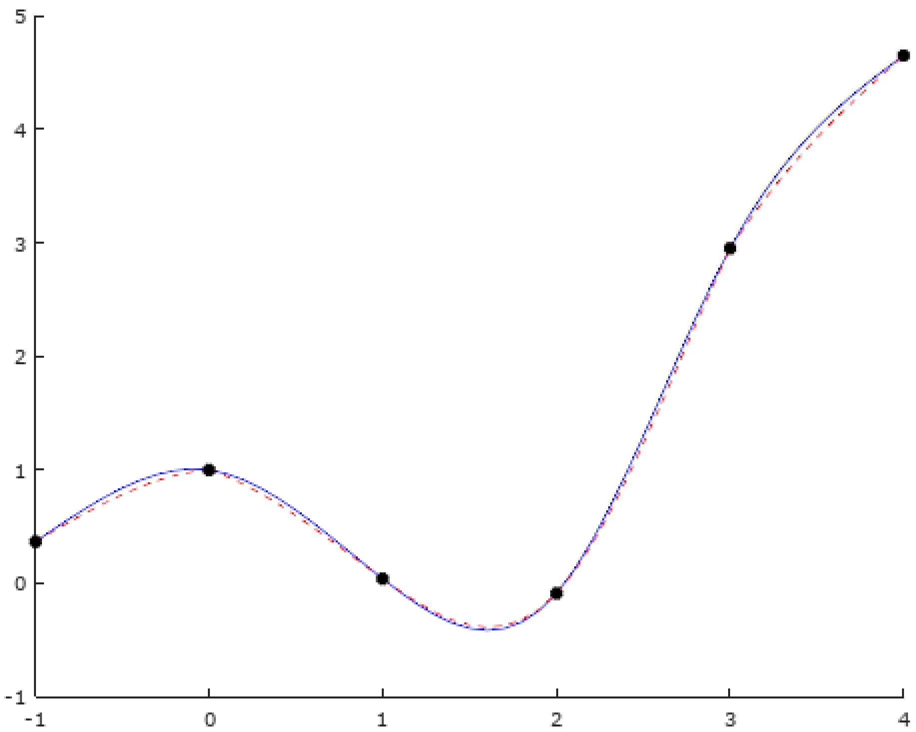
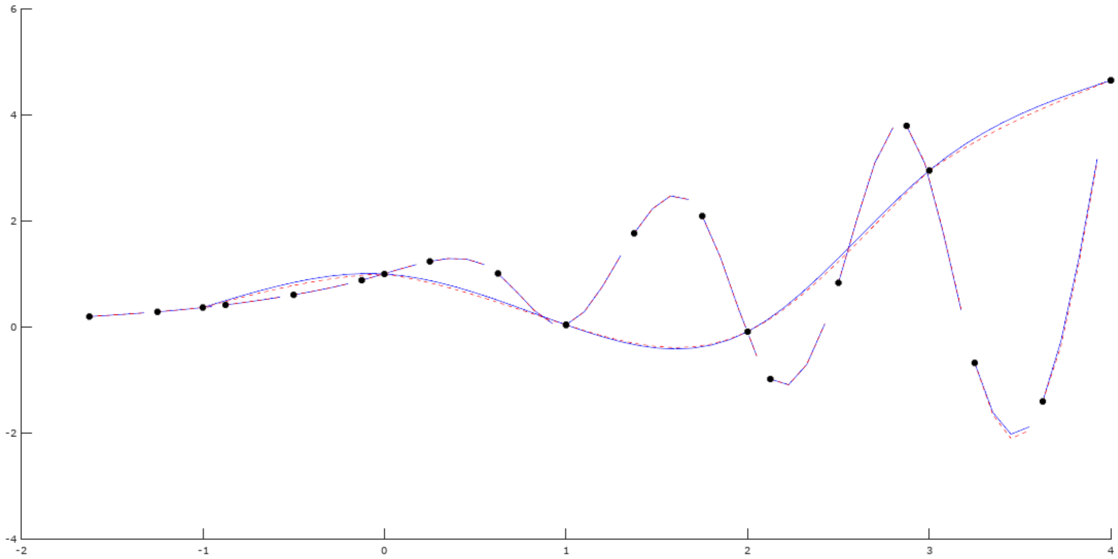


Figura 5 – Gráfico do *spline* natural e extrapolado para n = 17



Os resultados obtidos foram satisfatórios visto que ao traçar a função no *software Geogebra* voltado para desenhos de gráficos. Ferramenta muito útil na análise de gráficos. Os gráficos obtidos ficaram bastante próximos. As diferenças observáveis se dão devido a implementação e ao processo de compilação, bem como no sistema de numeração utilizado pelo código e pelo *software* que são distintos.

5 CONCLUSÃO

A aplicação de *splines* cúbicos para o desenho de gráficos são de grande importância, visto que o grau do polinômio resultante é limitado a 3, quando trata-se de um *spline* do terceiro grau. Em comparação aos métodos naturais em que o grau do polinômio para um grupo de n+1 pontos é n.

Além do mais os *splines* tendem a diminuir a mudar mudanças abruptas obtidas em polinômios de grau mais alto.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHAPRA, Steven C.; CANALE, Raymond P. **Métodos Numéricos para Engenharia**. 5. ed. São Paulo: AMGH, 2011.

[2] FRANCO, Neide Bertoldi. **Cálculo Númerico.** São Paulo: Pearson Prentice Hall, 2006.

[3] JUSTO, Dagoberto Adriano Rizzotto et al. **Cálculo Numérico Um Livro Colaborativo Versão Python.** Rio Grande do Sul: Ufrgs, 2018. Disponível em: <https://www.ufrgs.br/reamat/CalculoNumerico/livro-py/main.html>. Acesso em: 30 set. 2018.

[4] P., Carlos Armando de Castro. **Algoritmo en MATLAB para trazador cúbico.** Disponível em: <https://sites.google.com/site/matematicasingenieria/spline_orden_3>. Acesso em: 01 jan. 2018. (P., 2018).