Métodos Numéricos para Engenharia

MÓDULO 12- SPLINES - INTERPOLAÇÃO POR INTERVALOS

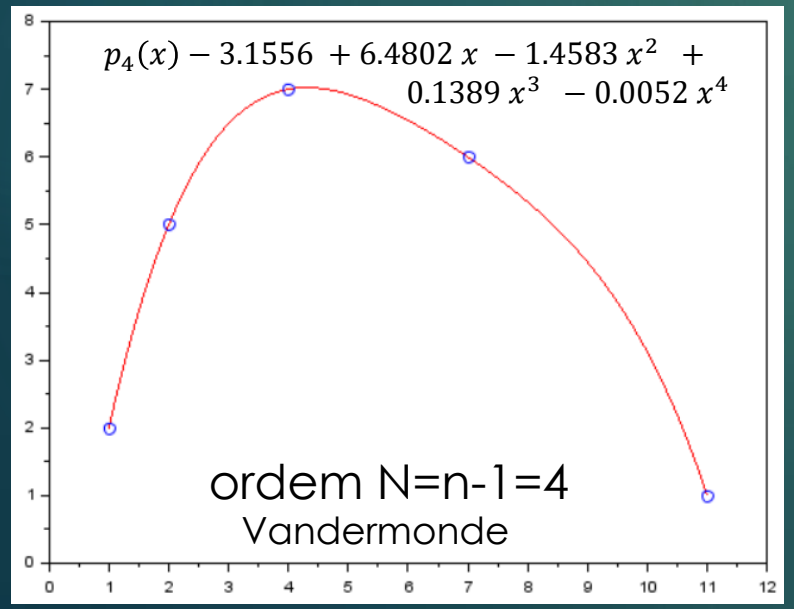
PROFESSOR LUCIANO NEVES DA FONSECA

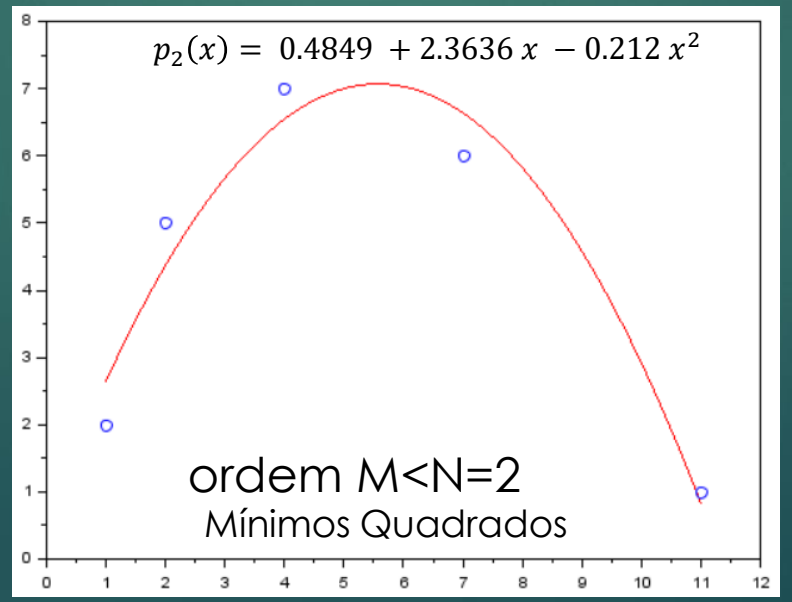
Splines

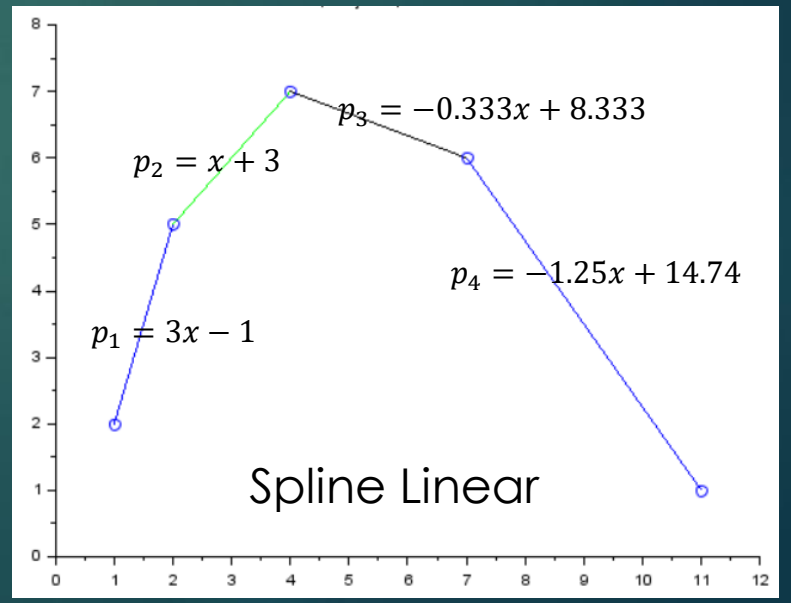
- Se tivermos N+1 pontos de controle, temos a opção de ajustarmos um polinômio interpolador de ordem N, que passa pelos N+1 pontos de controle. No entanto se N for muito grande, haverá oscilações (wiggling)
- Uma segunda opção seria ajustarmos um polinômio interpolador de ordem M<N, que passa 'entre' os pontos de controle, com o erro quadrático-médio minimizado. No entanto, como foi dito, este polinômio não passa pelos pontos de controle, o que pode ser um problema em algumas aplicações.
- Uma terceira opção seria ajustarmos um polinômio a cada um dos N intervalos $[x_i, x_{i+1}]$ entre os pontos de controle. Deste modo evitamos as oscilações (wiggling) e honramos os pontos de controle.
- O polinômio interpolador para cada intervalo pode ser Linear (Spline Linear), Quadrático (Spline Quadrática) ou Cúbico (Spline Cúbica).

n=5 pontos
N=4 intervalos

x_i	y_i
1	2
2	5
4	7
7	6
11	1





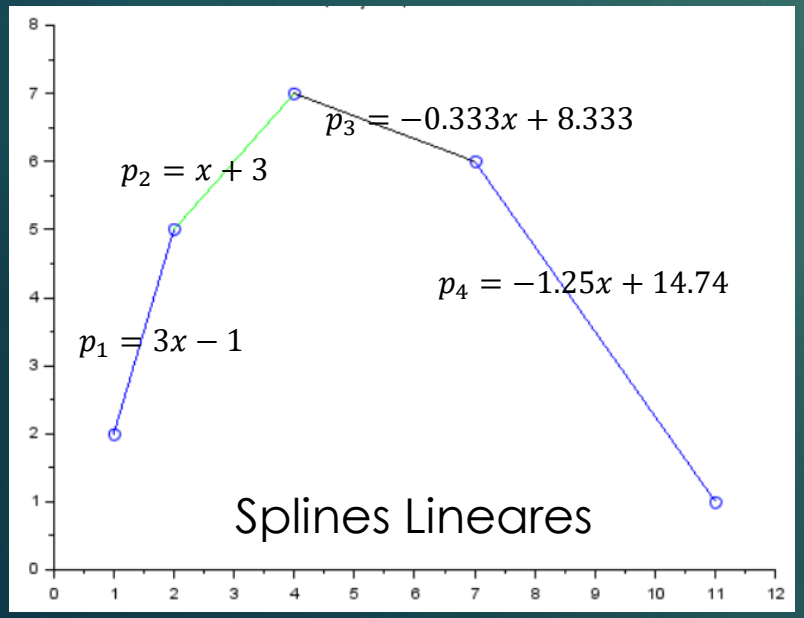


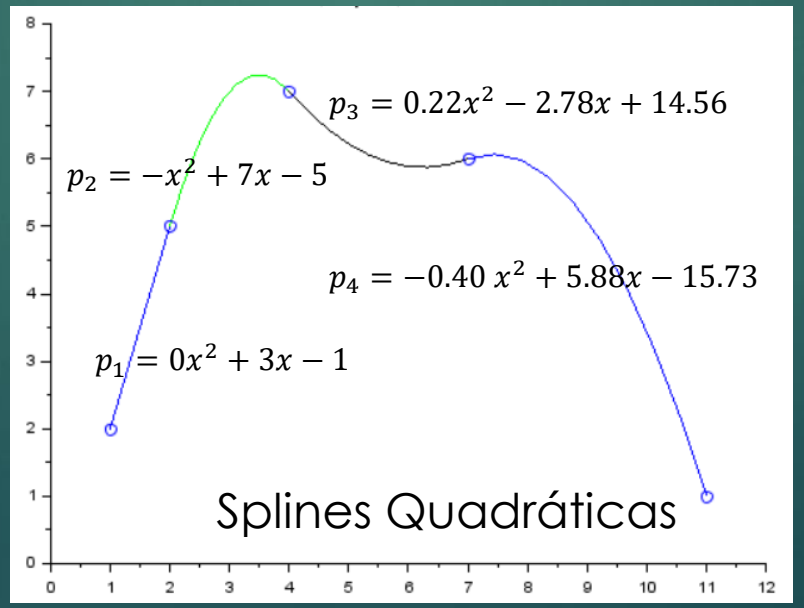
Splines Lineares, Quadráticas e Cúbicas

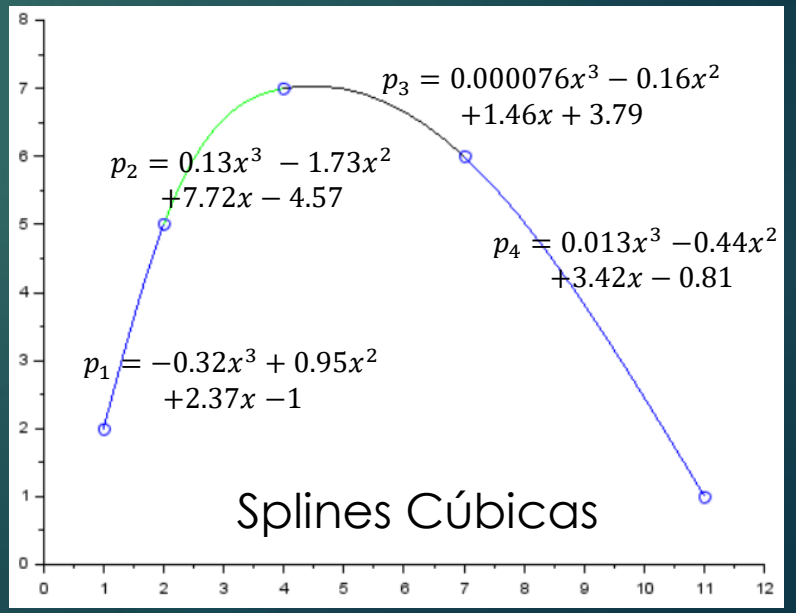
- Na interpolação por splines, o polinômio interpolador para cada intervalo pode ser Linear (Splines Lineares), Quadrático (Splines Quadráticas) ou Cúbico (Splines Cúbicas).
- Sem dúvida, o melhor resultado e o resultado mais consistente é o das Splines Cúbicas. Na prática esta são a mais utilizadas.
- Estudaremos as Splines Lineares e as Quadrática por uma questão didática, de forma a facilitar o aprendizado das Splines Cúbicas.

x_i	y_i
1	2
2	5
4	7
7	6
11	1

N=4







Splines Lineares

- Se tivermos N+1 pontos de controle, teremos N intervalos.
- Será ajustado uma reta (polinômio de ordem 1) a cada intervalo.
- O polinômio 1 irá unir o ponto $[x_1, y_1]$ ao ponto $[x_2, y_2]$ e o polinômio N o ponto $[x_N, y_N]$ ao ponto $[x_{N+1}, y_{N+1}]$
- Cada polinômio de ordem 1 acrescenta 2 incógnitas $p_n = a_n x + b_n$
- Como temos N polinômios, teremos 2N incógnitas.
- Precisamos de 2N equações, pois temos N intervalos, logo N polinômios.
- Cada polinômio deve honrar dois pontos de controle, o que nos fornece 2N equações.

n=5 pontos
N=4 intervalos

x_i	y_i
1	2
2	5
4	7
7	6
11	1

$$p_1(x_1) = a_1x_1 + b_1 = y_1$$
$$p_1(x_2) = a_1x_2 + b_1 = y_2$$
$$p_2(x_2) = a_2x_2 + b_2 = y_2$$
$$p_2(x_3) = a_2x_3 + b_2 = y_3$$
$$p_3(x_3) = a_3x_3 + b_3 = y_3$$
$$p_3(x_4) = a_3x_4 + b_3 = y_4$$
$$p_4(x_4) = a_4x_4 + b_4 = y_4$$
$$p_4(x_5) = a_4x_5 + b_4 = y_5$$

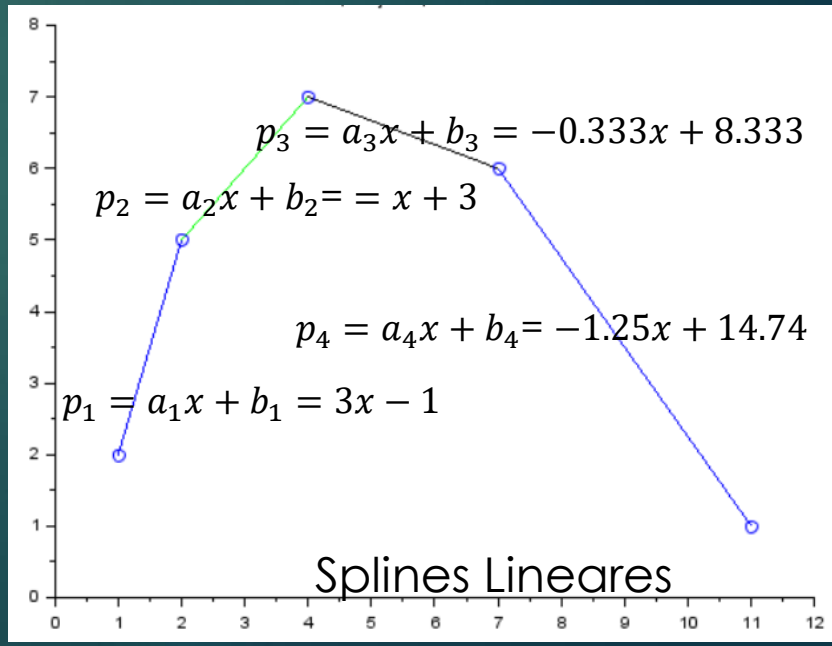
$$\begin{bmatrix} x_1 & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ x_2. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & x_2. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0 & x_3 & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & x_3. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & x_4. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & x_4. & 1. \\ 0. & 0. & 0. & 0. & 0. & 0. & x_5. & 1. \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \\ b_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_4 \\ y_5 \end{bmatrix}$$

$$Au = b$$

$$\begin{bmatrix} 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 2. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 2. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 4. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 4. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 7. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 7. & 1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 11. & 1. \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \\ b_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \\ 5 \\ 7 \\ 7 \\ 6 \\ 6 \\ 1 \end{bmatrix}$$

$$u = A^{-1}b$$

$$\begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \\ b_4 \end{bmatrix} = \begin{bmatrix} 3. \\ -1. \\ 1. \\ 3. \\ -0.33 \\ 8.33 \\ -1.25 \\ 14.75 \end{bmatrix}$$



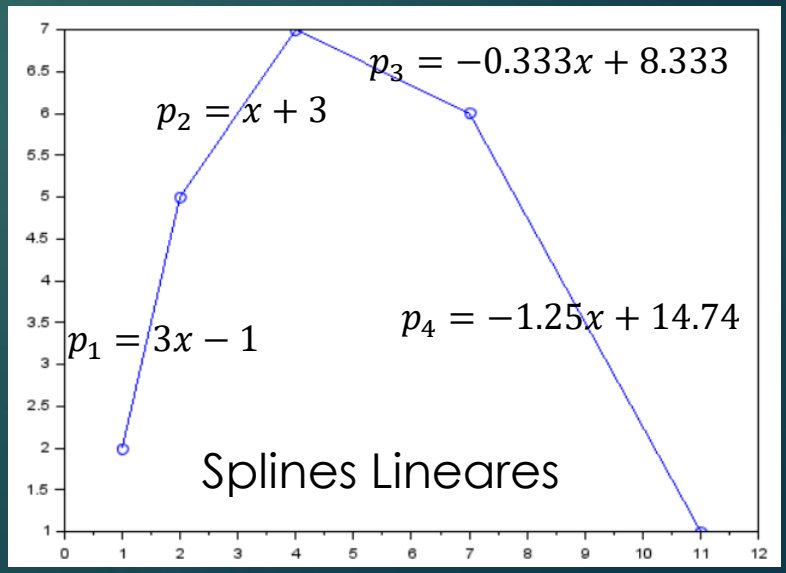
Notar que as Splines Lineares passam pelos pontos de controle
No entanto, a primeira derivada tem um descontinuidade nos pontos de controle

```
--> x=[1,2,4,7,11];
--> y=[2,5,7,6,1];
--> u=SplineLinear(x,y)
   3.
  -1.
   1.
   3.
  -0.3333333
   8.3333333
  -1.2500000
   14.750000
```

```
--> xp=linspace(min(x),max(x),1000);
--> plot(xp,InterSplineLinear(xp,x,u));
--> scatter(x,y)
```

```
function [u,A,b]=SplineLinear(x,y)
    s=poly(0,'s');
    N=length(x)-1 // # de intervalos
    b   =   zeros(2*N,1);
    A   =   zeros(2*N,2*N);
    index=1; //x(index),y(index)
    col=1;  // A(lin,col)
    //2N equações: funções passam por (xi,yi))
    for lin=1:2:2*N-1
        A(lin:lin+1,col:col+1)= [[x(index)   1];
                                 [x(index+1) 1]]
        b(lin:lin+1)=  [y(index);
                        y(index+1)] ;
        index=index+1;
        col=col+2;
    end
    u=tridiagonal(diag(A,-1),diag(A),diag(A,1),b)'
endfunction
```

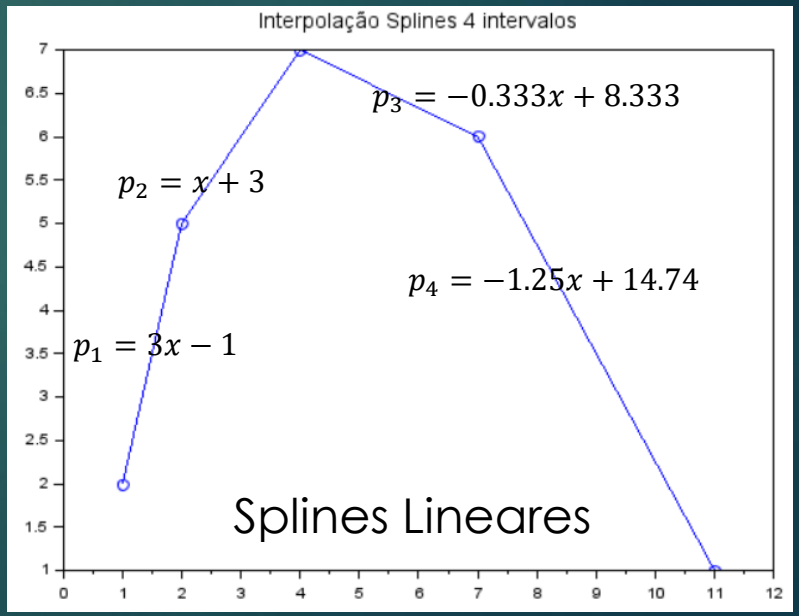
```
function yp=InterSplineLinear(xp,x,u)
    s=poly(0,'s');
    yp=xp*%nan;
    for (i=1:length(x)-1)
        index = find(xp>=x(i) & xp<=x(i+1));
        ps = poly([u(2*i),u(2*i-1)],"s","coeff")
        yp(index)=horner(ps,xp(index))
    end
endfunction
```



Notar que as Splines Lineares passam pelos pontos de controle
No entanto, a primeira derivada tem um descontinuidade nos pontos de controle

```
function Plot_SplineLinear(x,y)
    [u,A,b]=SplineLinear(x,y);
    for (i=1:length(x)-1)
      ps(i) = poly([u(2*i),u(2*i-1)],"s","coeff")
    end
    printf("Matriz Aumentada [A|b]")
    disp([A b]) // u = inv(A)*b
    disp(ps)
    N=length(x)-1 // # de intervalos
    xp=linspace(min(x),max(x),10000);
    plot(xp,InterSplineLinear(xp,x,u));
    scatter(x,y)
    xtitle("Interpolação Splines "+string(N)+" intervalos");
endfunction
```

```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> Plot_SplineLinear(x,y)
Matriz Aumentada [A|b]
   1.   1.   0.   0.   0.   0.   0.   0.   2.
   2.   1.   0.   0.   0.   0.   0.   0.   5.
   0.   0.   2.   1.   0.   0.   0.   0.   5.
   0.   0.   4.   1.   0.   0.   0.   0.   7.
   0.   0.   0.   0.   4.   1.   0.   0.   7.
   0.   0.   0.   0.   7.   1.   0.   0.   6.
   0.   0.   0.   0.   0.   0.   7.   1.   6.
   0.   0.   0.   0.   0.   0.   11.   1.   1.

  -1 +3s
  3 +s
  8.3333333 -0.3333333s
  14.75 -1.25s
```



Splines Quadráticas

- Se tivermos N+1 pontos de controle, teremos N intervalos.
- Será ajustado uma parábola (polinômio de ordem 2) a cada intervalo.
- O polinômio 1 irá unir o ponto $[x_1, y_1]$ ao ponto $[x_2, y_2]$ e o polinômio N o ponto $[x_N, y_N]$ ao ponto $[x_{N+1}, y_{N+1}]$
- Cada polinômio de ordem 2 acrescenta 3 incógnitas $p_n = a_n x^2 + b_n x + c_n$
- Como temos N polinômios, teremos 3N incógnitas.
- Precisamos de 3N equações, pois temos N intervalos, logo N polinômios.
- Cada polinômio deve honrar dois pontos de controle, o que nos fornece 2N equações.
- Para reduzir as descontinuidades, igualamos as derivadas nos pontos internos (N-1 equações)
- Para a última equação fazemos a1=0 (o primeiro polinômio será uma reta")

n=5 pontos
N=4 intervalos

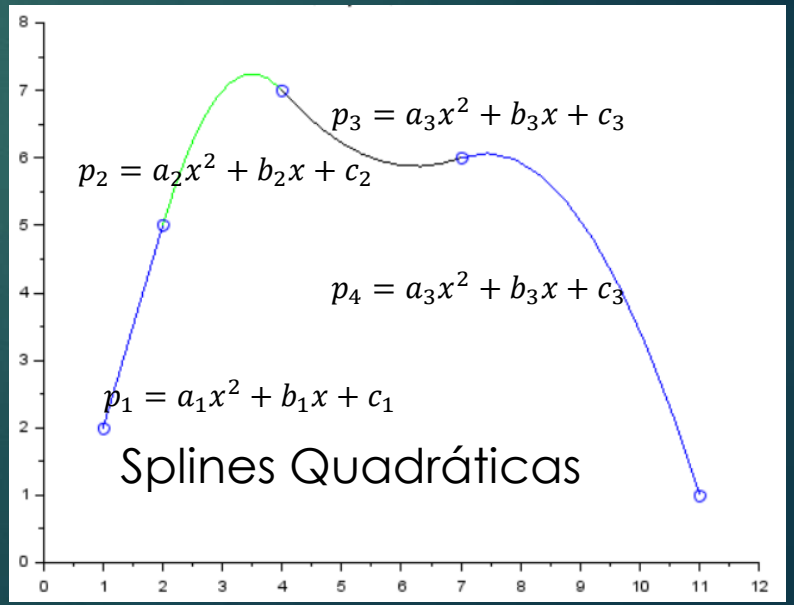
x_i	y_i
1	2
2	5
4	7
7	6
11	1

$$a_1 = 0$$

$$p_1(x_1) = a_1 x_1^2 + b_1 x_1 + c_1 = y_1$$
$$p_1(x_2) = a_1 x_2^2 + b_1 x_2 + c_1 = y_2$$
$$p_2(x_2) = a_2 x_2^2 + b_2 x_2 + c_2 = y_2$$
$$p_2(x_3) = a_2 x_3^2 + b_2 x_3 + c_2 = y_3$$
$$p_3(x_3) = a_3 x_3^2 + b_3 x_3 + c_3 = y_3$$
$$p_3(x_4) = a_3 x_4^2 + b_3 x_4 + c_3 = y_4$$
$$p_4(x_4) = a_4 x_4^2 + b_4 x_4 + c_4 = y_4$$
$$p_4(x_5) = a_4 x_5^2 + b_4 x_5 + c_4 = y_5$$
$$2a_1 x_2 + b_1 - 2a_2 x_2 - b_2 = 0$$
$$2a_2 x_3 + b_2 - 2a_3 x_3 - b_3 = 0$$
$$2a_3 x_4 + b_3 - 2a_4 x_4 - b_4 = 0$$

$$Au = b$$

$$\begin{bmatrix}
1 & 0. & 0. & 0. & 0. & 0. & 0. & 0.. & 0. & 0. & 0. & 0. \\
x_1^2 & x_1 & 1 & 0. & 0. & 0. & 0. & 0.. & 0. & 0. & 0. & 0. \\
x_2^2 & x_2 & 1 & 0. & 0. & 0. & 0. & 0.. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & x_2^2 & x_2 & 1 & 0. & 0.. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & x_3^2 & x_3 & 1 & 0. & 0.. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & x_3^2 & x_3 & 1 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & x_4^2 & x_4 & 1 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & x_4^2 & x_4 & 1 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & x_5^2 & x_5 & 1 \\
2x_2 & 1. & 0. & -2x_2 & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0 \\
0. & 0. & 0. & 2x_3 & 1. & 0. & -2x_3 & -1. & 0. & 0. & 0. & 0 \\
0. & 0. & 0. & 0. & 0. & 0. & 2x_4 & 1. & 0. & -2x_4 & -1. & 0
\end{bmatrix}
\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix}
=
\begin{bmatrix} 0 \\ y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_4 \\ y_5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



Splines Quadráticas

n=5 pontos
N=4 intervalos

x_i	y_i
1	2
2	5
4	7
7	6
11	1

$$a_1 = 0$$

$$p_1(x_1) = a_1x_1^2 + b_1x_1 + c_1 = y_1$$
$$p_1(x_2) = a_1x_2^2 + b_1x_2 + c_1 = y_2$$
$$p_2(x_2) = a_2x_2^2 + b_2x_2 + c_2 = y_2$$
$$p_2(x_3) = a_2x_3^2 + b_2x_3 + c_2 = y_3$$
$$p_3(x_3) = a_3x_3^2 + b_3x_3 + c_3 = y_3$$
$$p_3(x_4) = a_3x_4^2 + b_3x_4 + c_3 = y_4$$
$$p_4(x_4) = a_4x_4^2 + b_4x_4 + c_4 = y_4$$
$$p_4(x_5) = a_4x_5^2 + b_4x_5 + c_4 = y_5$$
$$2a_1x_2 + b_1 - 2a_2x_2 - b_2 = 0$$
$$2a_2x_3 + b_2 - 2a_3x_3 - b_3 = 0$$
$$2a_3x_4 + b_3 - 2a_4x_4 - b_4 = 0$$

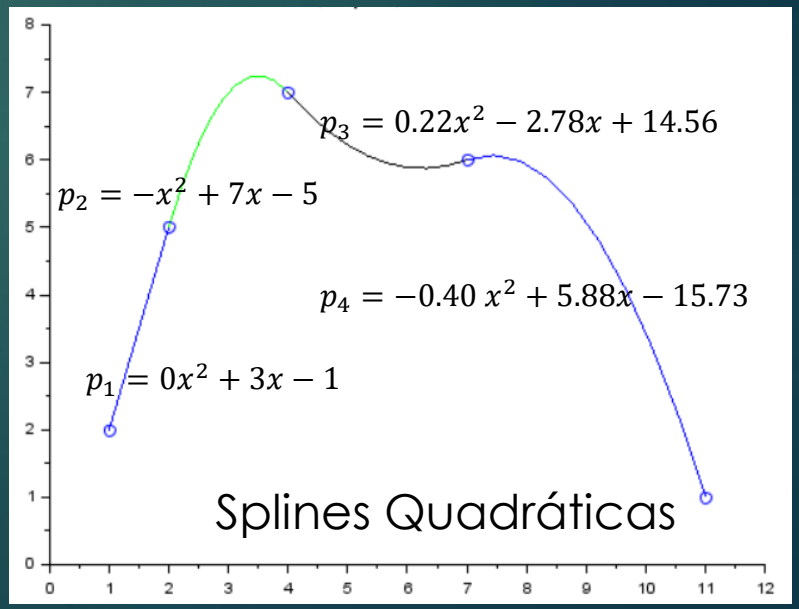
$$\begin{bmatrix} 1 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ x_1^2 & x_1 & 1 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ x_2^2 & x_2 & 1 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & x_2^2 & x_2 & 1 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & x_3^2 & x_3 & 1 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & x_3^2 & x_3 & 1 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & x_4^2 & x_4 & 1 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & x_4^2 & x_4 & 1 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & x_5^2 & x_5 & 1 \\ 2x_2 & 1. & 0. & -2x_2 & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0 \\ 0. & 0. & 0. & 2x_3 & 1. & 0. & -2x_3 & -1. & 0. & 0. & 0. & 0 \\ 0. & 0. & 0. & 0. & 0. & 0. & 2x_4 & 1. & 0. & -2x_4 & -1. & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0 \\ y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_4 \\ y_5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Au = b$$

$$\begin{bmatrix} 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 4. & 2. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 4. & 2. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 16. & 4. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 16. & 4. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 49. & 7. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 49. & 7. & 1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 121. & 11. & 1. \\ 4. & 1. & 0. & -4. & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 8. & 1. & 0. & -8. & -1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 14. & 1. & 0. & -14. & -1. & 0. \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0. \\ 2. \\ 5. \\ 5. \\ 7. \\ 7. \\ 6. \\ 6. \\ 1. \\ 0. \\ 0. \\ 0. \end{bmatrix} \begin{matrix} 0 \\ y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_4 \\ y_5 \\ 0 \\ 0 \\ 0 \end{matrix}$$

$$u = A^{-1}b$$

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0. \\ 3. \\ -1. \\ -1. \\ 7. \\ -5. \\ 0.222 \\ -2.778 \\ 14.556 \\ -0.396 \\ 5.875 \\ -15.729 \end{bmatrix}$$



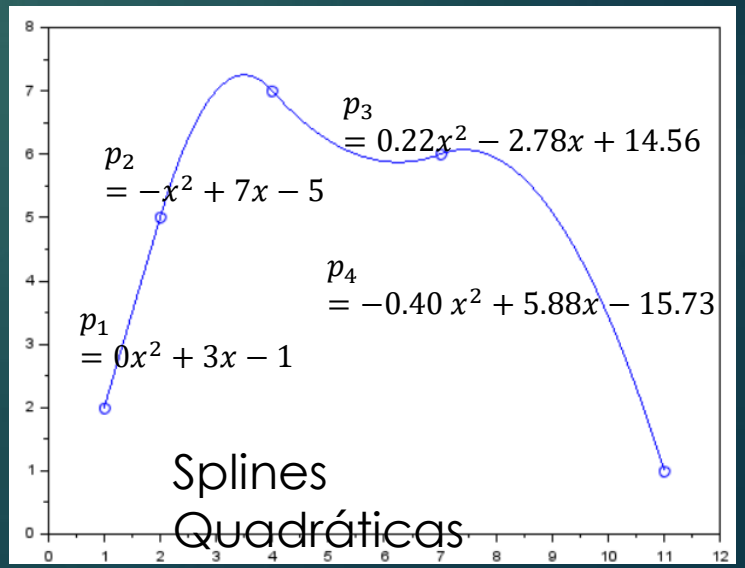
```
function [u,A,b]=SplineQuadratica(x,y)
    N=length(x)-1  // # de intervalos
    b  =  zeros(3*N,1);
    A  =  zeros(3*N,3*N);
    index=1; //x(index),y(index)
    col=1; // A(lin,col)
    A(1,1)=1; // 1a equação a1=0
    for lin=2:2:2*N  //2N equações: funções passam por (xi,yi))
        A(lin:lin+1,col:col+2)= [[x(index)^2  x(index)  1];
                                 [x(index+1)^2 x(index+1) 1]]
        b(lin:lin+1)=  [y(index);
                        y(index+1)] ;
        index=index+1;
        col=col+3;
    end
    col=1;
    index=2;
    for lin=2*N+2:3*N //N-1 equações-1a derivadas pontos internos
        A(lin,col:col+5)=  [2*x(index) 1 0 -2*x(index) -1 0];
        index=index+1;
        col=col+3;
    end
    u=EliminacaoGaussJordan(A,b)
endfunction
```

```
function yp=InterSplineQuadratica(xp,x,u)
  s=poly(0,'s');
  yp=xp*%nan;
  for (i=1:length(x)-1)
    index = find(xp>=x(i) & xp<=x(i+1));
    ps = poly([u(3*i),u(3*i-1),u(3*i-2)],"s","coeff")
    yp(index)=horner(ps,xp(index))
  end
endfunction
```

```
--> x=[1,2,4,7,11];
--> y=[2,5,7,6,1];
--> u=SplineQuadratica(x,y)

   0.
   3.
  -1.
  -1.
   7.
  -5.
   0.2222222
  -2.7777778
   14.555556
  -0.3958333
   5.8750000
  -15.729167
```

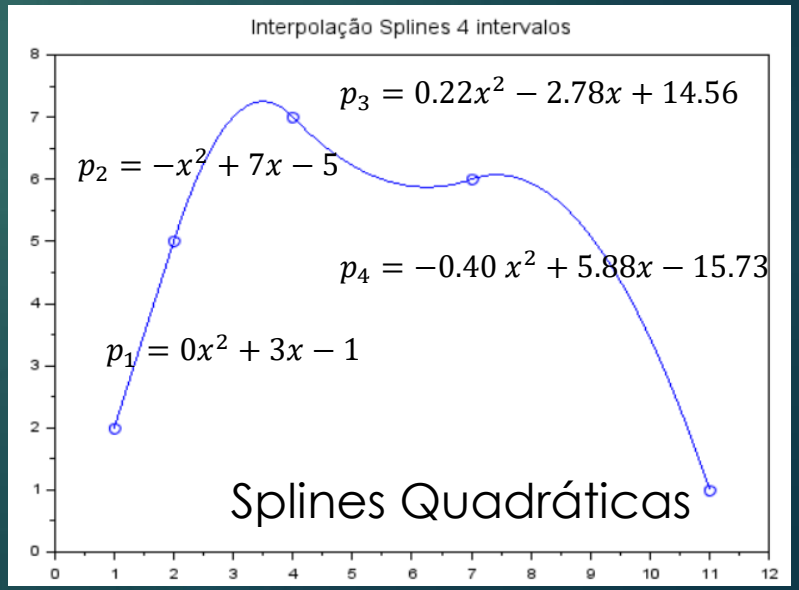
```
--> xp=linspace(min(x),max(x),1000);

--> plot(xp,InterSplineQuadratica(xp,x,u));

--> scatter(x,y)
```

- Notar que o 1° polinômio é linear!
- Há inflexões desnecessárias nos pontos de controle



```
1  function Plot_SplineQuadratica(x,y)
2      [u,A,b]=SplineQuadratica(x,y);
3      for (i=1:length(x)-1)
4          ps(i) = poly([u(3*i),u(3*i-1),u(3*i-2)],"s","coeff")
5      end
6      printf("Matriz Aumentada [A|b]")
7      disp([A b]) // u = inv(A)*b
8      disp(ps)
9      N=length(x)-1 // # de intervalos
10     xp=linspace(min(x),max(x),10000);
11     plot(xp,InterSplineQuadratica(xp,x,u));
12     scatter(x,y)
13     xtitle("Interpolação Splines "+string(N)+" intervalos");
14 endfunction
```

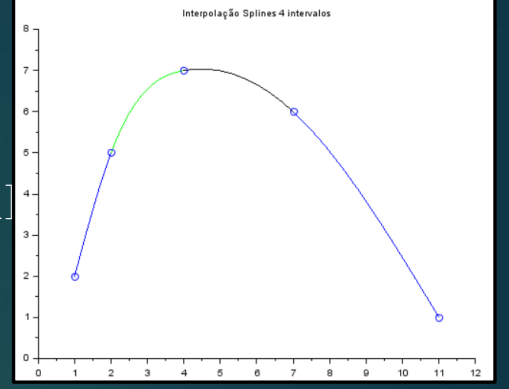
```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> Plot_SplineQuadratica(x,y)
Matriz Aumentada [A|b]
   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
   1.   1.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   2.
   4.   2.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   5.
   0.   0.   0.   4.   2.   1.   0.   0.   0.   0.   0.   0.   5.
   0.   0.   0.   16.   4.   1.   0.   0.   0.   0.   0.   0.   7.
   0.   0.   0.   0.   0.   0.   16.   4.   1.   0.   0.   0.   7.
   0.   0.   0.   0.   0.   0.   49.   7.   1.   0.   0.   0.   6.
   0.   0.   0.   0.   0.   0.   0.   0.   0.   49.   7.   1.   6.
   0.   0.   0.   0.   0.   0.   0.   0.   0.   121.   11.   1.   1.
   4.   1.   0.   -4.   -1.   0.   0.   0.   0.   0.   0.   0.   0.
   0.   0.   0.   8.   1.   0.   -8.   -1.   0.   0.   0.   0.   0.
   0.   0.   0.   0.   0.   0.   14.   1.   0.   -14.   -1.   0.   0.

  -1 +3s
  -5 +7s -s²
  14.555556 -2.7777778s +0.2222222s²
  -15.729167 +5.875s -0.3958333s²
```



- Notar que o 1° polinômio é linear!
- Há inflexões desnecessárias nos pontos de controle

Splines Cúbicas

n=5 pontos
N=4 intervalos



- Se tivermos N+1 pontos de controle, teremos N intervalos.
- Será ajustado polinômio de ordem 3 a cada intervalo.
- O polinômio 1 irá unir o ponto $[x_1, y_1]$ ao ponto $[x_2, y_2]$ e o polinômio N, do ponto $[x_N, y_N]$ ao ponto $[x_{N+1}, y_{N+1}]$
- Cada polinômio de ordem 3 acrescenta 4 incógnitas $p_n = a_n x^3 + b_n x^2 + c_n x + d_n$
- Como temos N polinômios, teremos 4N incógnitas.
- Precisamos de 4N equações, pois temos N intervalos (N polinômios).
- Cada polinômio deve honrar dois pontos de controle, o que nos fornece 2N equações.
- Para reduzir as descontinuidades, igualamos as derivadas nos pontos internos (N-1 equações)
- Para reduzir as inflexões , igualamos as segundas derivadas nos pontos internos (N-1 equações)
- Para a duas últimas equação anulamos a segunda derivadas no primeiro e no ultimo ponto (spline natural)

$$p_1(x_1) = a_1x_1^3 + b_1x_1^2 + c_1x_1 + d_1 = y_1$$
$$p_1(x_2) = a_1x_2^3 + b_1x_2^2 + c_1x_2 + d_1 = y_2$$
$$p_2(x_2) = a_2x_2^3 + b_2x_2^2 + c_2x_2 + d_2 = y_2$$
$$p_2(x_3) = a_2x_3^3 + b_2x_3^2 + c_2x_3 + d_2 = y_3$$
$$p_3(x_3) = a_3x_3^3 + b_3x_3^2 + c_3x_3 + d_3 = y_3$$
$$p_3(x_4) = a_3x_4^3 + b_3x_4^2 + c_3x_4 + d_3 = y_4$$
$$p_4(x_4) = a_4x_4^3 + b_4x_4^2 + c_4x_4 + d_4 = y_4$$
$$p_4(x_5) = a_4x_5^3 + b_4x_5^2 + c_4x_5 + d_4 = y_5$$
$$3a_1x_2^2 + 2b_1x_2 + c_1 = 3a_2x_2^2 + 2b_2x_2 + c_2$$
$$3a_2x_3^2 + 2b_2x_3 + c_2 = 3a_3x_3^2 + 2b_3x_3 + c_3$$
$$3a_3x_4^2 + 2b_3x_4 + c_3 = 3a_4x_4^2 + 2b_4x_4 + c_4$$
$$6a_1x_1 + 2b_1 = 0$$
$$6a_1x_2 + 2b_1 = 6a_2x_2 + 2b_2$$
$$6a_2x_3 + 2b_2 = 2a_3x_3 + 2b_3$$
$$6a_3x_4 + 2b_3 = 2a_4x_4 + 2b_4$$
$$6a_3x_5 + 2b_3 = 0$$

$$\begin{bmatrix}
x_1^3 & x_1^2 & x_1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_2^3 & x_2^2 & x_2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_2^3 & x_2^2 & x_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_3^3 & x_3^2 & x_3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_3^3 & x_3^2 & x_3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_4^3 & x_4^2 & x_3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_4^3 & x_4^2 & x_3 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_5^3 & x_5^2 & x_5 & 1 \\
3x_2^2 & 2x_2 & 1 & 0 & -3x_2^2 - 2x_2 - 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3x_3^2 & 2x_3 & 1 & 0 & -3x_3^2 - 2x_3 - 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3x_4^2 & 2x_4 & 1 & 0 & -3x_4^2 - 2x_4 - 1 & 0 & 0 & 0 \\
6x_1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6x_2 & 2 & 0 & 0 & -6x_2 - 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6x_3 & 2 & 0 & 0 & -6x_3 - 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 6x_4 & 2 & 0 & 0 & -6x_4 - 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 6x_5 & 2 & 0 & 0
\end{bmatrix}
\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \\ a_3 \\ b_3 \\ c_3 \\ d_3 \\ a_4 \\ b_4 \\ c_4 \\ d_4 \end{bmatrix}
=
\begin{bmatrix} y_1 \\ y_2 \\ y_2 \\ y_3 \\ y_3 \\ y_4 \\ y_4 \\ y_5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

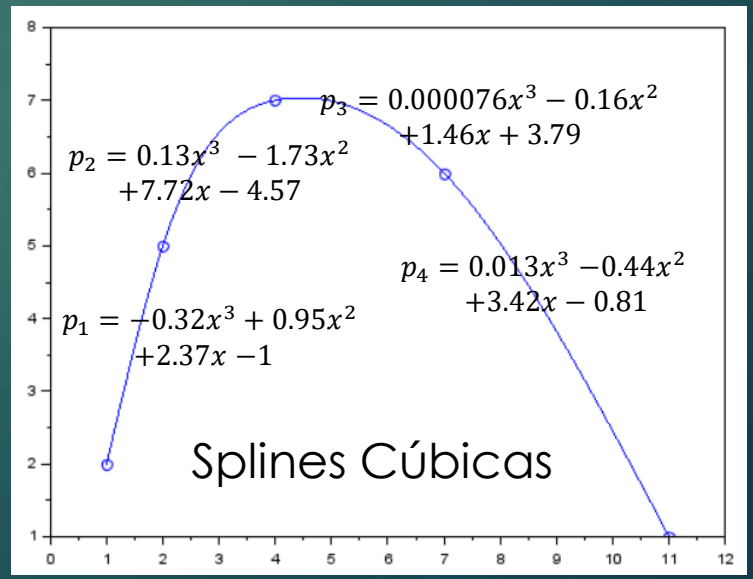
x_i	y_i
1	2
2	5
4	7
7	6
11	1

```
function [u,A,b]=SplineCubica(x,y)
    s=poly(0,'s');
    N=length(x)-1  // # de intervalos
    b    =    zeros(4*N,1);
    A    =    zeros(4*N,4*N);
    index=1; //x(index),y(index)
    col=1;
    for lin=1:2:2*N-1      //2N equações: funções passam por (xi,yi))
        A(lin:lin+1,col:col+3)= [[x(index)^3    x(index)^2    x(index)    1];
                                 [x(index+1)^3 x(index+1)^2 x(index+1) 1]]
        b(lin:lin+1)=  [y(index);
                        y(index+1)] ;
        index=index+1;
        col=col+4;
    end
    col=1;
    index=2;
    for lin=2*N+1:3*N-1    // N-1 equações - 1a derivadas pontos internos
        A(lin,col:col+7)=[3*x(index)^2  2*x(index)  1  0  -3*x(index)^2  -2*x(index)  -1  0];
        index=index+1;
        col=col+4;
    end
    col=1;
    index=1;
    A(3*N,col:col+3)=    [6*x(index)    2    0  0]; //2a derivada ponto externo
    index=index+1;
    for lin=3*N+1:4*N-1    // N-1 equações - 2a derivadas pontos internos
        A(lin,col:col+7)=    [6*x(index)    2    0  0  -6*x(index)  -2  0  0];
        index=index+1;
        col=col+4;
    end
    A(4*N,col:col+3)=    [6*x(index)    2    0  0];//2a derivada ponto externo
    u=EliminacaoGaussJordan(A,b)
endfunction
```

```
function yp=InterSplineCubica(xp,x,u)
    s=poly(0,'s');
    yp=xp*%nan;
    for (i=1:length(x)-1)
        index = find(xp>=x(i) & xp<=x(i+1));
        ps = poly([u(4*i),u(4*i-1),u(4*i-2),u(4*i-3)],"s","coeff")
        yp(index)=horner(ps,xp(index))
    end
endfunction
```

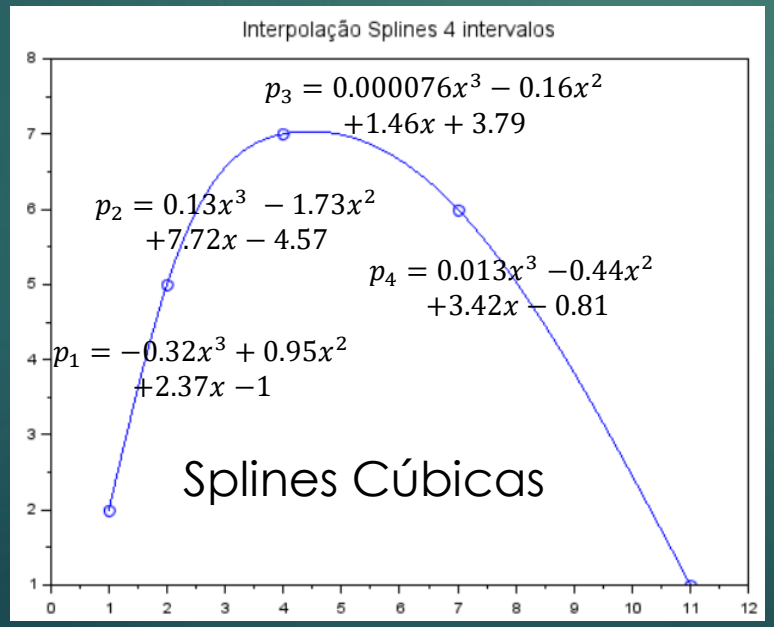
```
--> x=[1,2,4,7,11];
--> y=[2,5,7,6,1];
--> u=SplineCubica(x,y)
  -0.315296804
   0.945890411
   2.369406393
  -1.000000000
   0.130593607
  -1.729452055
   7.720091324
  -4.567123288
   0.000076104
  -0.163242009
   1.455251142
   3.785996956
   0.013470320
  -0.444520548
   3.424200913
  -0.808219178
```

```
--> xp=linspace(min(x),max(x),1000);
--> plot(xp,InterSplineCubica(xp,x,u));
--> scatter(x,y)
```



```
function Plot_SplineCubica(x,y)
    [u,A,b]=SplineCubica(x,y);
    for (i=1:length(x)-1)
        ps(i) = poly([u(4*i),u(4*i-1),u(4*i-2),u(4*i-3)],"s","coeff")
    end
    printf("Matriz Aumentada [A|b]")
    disp([A b]) // u = inv(A)*b
    disp(ps)
    N=length(x)-1 // # de intervalos
    xp=linspace(min(x),max(x),10000);
    plot(xp,InterSplineCubica(xp,x,u));
    scatter(x,y)
    xtitle("Interpolação Splines "+string(N)+" intervalos");
endfunction
```

```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> Plot_SplineCubica(x,y)
Matriz Aumentada [A|b]
   1.    1.    1.    1.    0.    0.    0.    0.    0.     0.    0.    0.    0.      0.     0.    0.    2.
   8.    4.    2.    1.    0.    0.    0.    0.    0.     0.    0.    0.    0.      0.     0.    0.    5.
   0.    0.    0.    0.    8.    4.    2.    1.    0.     0.    0.    0.    0.      0.     0.    0.    5.
   0.    0.    0.    0.    64.   16.   4.    1.    0.     0.    0.    0.    0.      0.     0.    0.    7.
   0.    0.    0.    0.    0.    0.    0.    0.    64.    16.   4.    1.    0.      0.     0.    0.    7.
   0.    0.    0.    0.    0.    0.    0.    0.    343.   49.   7.    1.    0.      0.     0.    0.    6.
   0.    0.    0.    0.    0.    0.    0.    0.    0.     0.    0.    0.    343.    49.    7.    1.    6.
   0.    0.    0.    0.    0.    0.    0.    0.    0.     0.    0.    0.    1331.   121.   11.   1.    1.
   12.   4.    1.    0.   -12.  -4.   -1.   0.    0.     0.    0.    0.    0.      0.     0.    0.    0.
   0.    0.    0.    0.    48.   8.    1.    0.   -48.   -8.   -1.   0.    0.      0.     0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    147.   14.   1.    0.   -147.   -14.   -1.   0.    0.
   6.    2.    0.    0.    0.    0.    0.    0.    0.     0.    0.    0.    0.      0.     0.    0.    0.
   12.   2.    0.    0.   -12.  -2.    0.    0.    0.     0.    0.    0.    0.      0.     0.    0.    0.
   0.    0.    0.    0.    24.   2.    0.    0.   -24.   -2.    0.    0.    0.      0.     0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    42.    2.    0.    0.   -42.    -2.    0.    0.    0.
   0.    0.    0.    0.    0.    0.    0.    0.    0.     0.    0.    0.    66.     2.    0.    0.    0.

  -1 +2.3694064s +0.9458904s² -0.3152968s³
  -4.5671233 +7.7200913s -1.7294521s² +0.1305936s³
  3.785997 +1.4552511s -0.163242s² +0.0000761s³
  -0.8082192 +3.4242009s -0.4445205s² +0.0134703s³
```



- Notar que as Splines cúbicas oferecem, sem dúvida, a melhor interpolação, sem descontinuidade e sem inflexões
- No entanto a matriz A é esparsa e de ordem muito alta, o que pode inviabilizar o método se N é muito grande

Cálculo Alternativo para Splines Naturais Cúbicas

(nas Splines naturais, as segundas derivadas nos pontos externos são nulas)

n=5 pontos
N=4 intervalos

x_i	y_i
1	2
2	5
4	7
7	6
11	1

$$p_i(x) = a_i + b_i(x - x_i) + c_i\,(x - x_i)^2 + d_i\,(x - x_i)^3$$

O polinômio cúbico terá um forma alternativa, de modo que as incógnitas serão os novos coeficientes $a_i, b_i, c_i \; e \; d_i$ (e não os coeficientes u das splines cúbicas)

$a_i = p_i(x_i) = y_i$ $\quad a_i$ é o *valor do polinônio* $p_i(x)$ *no início do intervalo*

$b_i = p'_i(x_i)$ $\quad b_i$ é o *valor* $p'_i(x)$ *no início do intervalo*

$p'_i(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$

$c_i = \frac{1}{2}p''_i(x_i)$ $\quad c_i$ é a *metade do valor* $p''_i(x)$ *no início do intervalo*

$p''_i(x) = 2c_i + 6d_i(x - x_i)$

Os coeficientes $a_i, b_i, c_i \; e \; d_i$ podem ser calculados através de uma tabela de diferenças:

Algoritmo de spline natural:

- O algoritmo de spline natural constrói um spline que tem derivadas de segundo zero nos limites.
- O algoritmo envolve a resolução de um sistema linear tridiagonal para determinar os coeficientes do spline.
- O sistema linear tridiagonal é formado reforçando a continuidade da primeira e segunda derivadas em cada nó interior.
- As condições de contorno são definidas fixando as segundas derivadas em zero no primeiro e no último nós.
- A spline resultante é suave e tem um comportamento natural nos limites.

Tabela de Diferenças

i	x_i	$h=x_{i+1}-x_i$	y_i	$dy=(y_{i+1}-y_i)/(x_{i+1}-xi)$	$r=3(dy_i-dy_{i-1})$	$2*(h_i+h_{i-1})$
1	1.00	1.00	2.00	3.00		
2	2.00	2.00	5.00	1.00	-6.00	6.00
3	4.00	3.00	7.00	-0.33	-4.00	10.00
4	7.00	4.00	6.00	-1.25	-2.75	14.00
5	11.00		1.00			

n=5 pontos
N=4 intervalos

$$p_i(x) = a_i + b_i(x - x_i) + c_i\,(x - x_i)^2 + d_i\,(x - x_i)^3$$

x_i	y_i
1	2
2	5
4	7
7	6
11	1

Tabela de Diferenças

i	x_i	$h=x_{i+1}-x_i$	y_i	$dy=(y_{i+1}-y_i)/(x_{i+1}-xi)$	$r=3(dy_i-dy_{i-1})$	$2*(h_i+h_{i-1})$
1	1.00	1.00	2.00	3.00		
2	2.00	2.00	5.00	1.00	-6.00	6.00
3	4.00	3.00	7.00	-0.33	-4.00	10.00
4	7.00	4.00	6.00	-1.25	-2.75	14.00
5	11.00		1.00			

A partir da tabela de Tabela de Diferenças, podemos montar um sistema de equações tridiagonal (N-1)x(N-1) para calcular c_i , que são as segundas derivadas nos pontos internos (lembrar que no pontos externos $c_o = c_N = 0;$)

Sistema e Equações

i=2	$2*(h_1+h_2)$	h_2			c_2		r_2
i=3	h_2	$2*(h_2+h_3)$	h_3	*	c_3	=	r_3
i=4		h_3	$2*(h_3+h_4)$		c_4		r_4
	6	2			c_2		-6
	2	10	3	*	c_3	=	-4
		3	14		c_4		-2.75

Resolver Sistema

-0.946
-0.162
-0.162

Com c_i podemos calcular b_i e d_i $(a_i = y_i)$

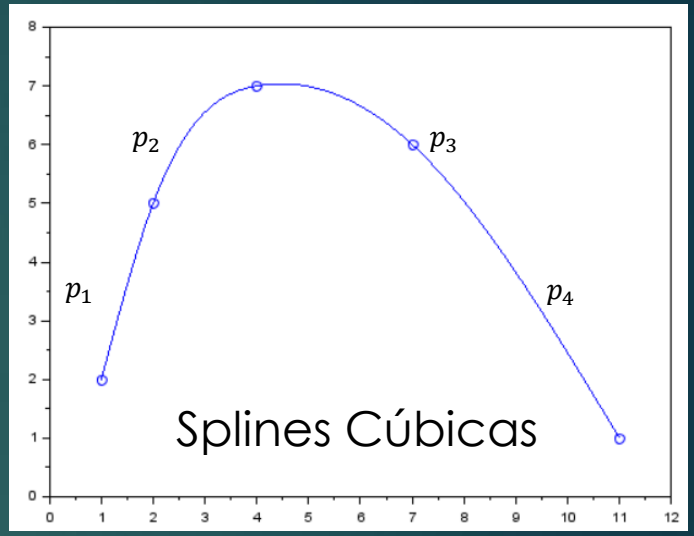
i	$a_i=y_i$	bi $=dy_i-h_i/3*(c_{i+1}+2c_i)$	c_i	$d_i=(c_{i+1}-ci)/(3*h_i)$
1	2.00	3.31530	0.00000	-0.31530
2	5.00	2.36941	-0.94589	0.13059
3	7.00	0.15297	-0.16233	0.00008
4	6.00	-0.81895	-0.16164	0.01347

```
function [a,b,c,d]=SplineNatural(x,y)
    N=length(x)-1 // # de intervalos
    h  =diff(x);
    dy =diff(y)./h;
    dl=h(2:N-1);
    dp=[2*(h(1:N-1)+h(2:N))];
    du=h(2:N-1);
    r=3*diff(dy)
    c=[0 tridiagonal(dl,dp,du,r) 0]';
    a=y';
    for (k=1:N)
      b(k)= dy(k)-h(k)*(2*c(k)+c(k+1))/3;
      d(k)= (c(k+1)-c(k))/(3*h(k));
    end
endfunction
```

```
function yp=InterSplineNatural(xp,x,a,b,c,d)
 s=poly(0,'s');
 yp=xp*%nan;
 for (i=1:length(x)-1)
    index = find(xp>=x(i) & xp<=x(i+1));
    ps_temp=poly([a(i),b(i),c(i),d(i)],"s","coeff")
    ps=horner(ps_temp,(s-x(i)))
    yp(index)=horner(ps,xp(index))
 end
endfunction
```

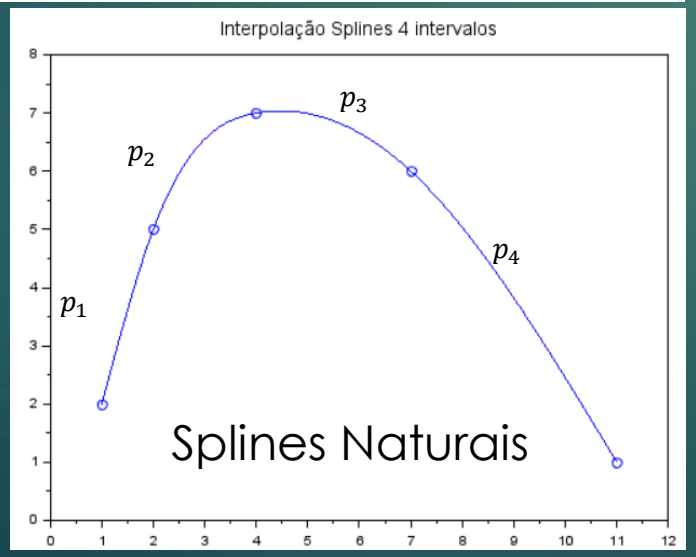
```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> [a,b,c,d]=SplineNatural(x,y)
 a  =

   2.
   5.
   7.
   6.
   1.
 b  =

   3.3152968
   2.3694064
   0.1529680
  -0.8189498
 c  =

   0.
  -0.9458904
  -0.1623288
  -0.1616438
   0.
 d  =

  -0.3152968
   0.1305936
   0.0000761
   0.0134703
```

```
--> xp=linspace(min(x),max(x),1000);

--> plot(xp,InterSplineNatural(xp,x,a,b,c,d))

--> scatter(x,y)
```



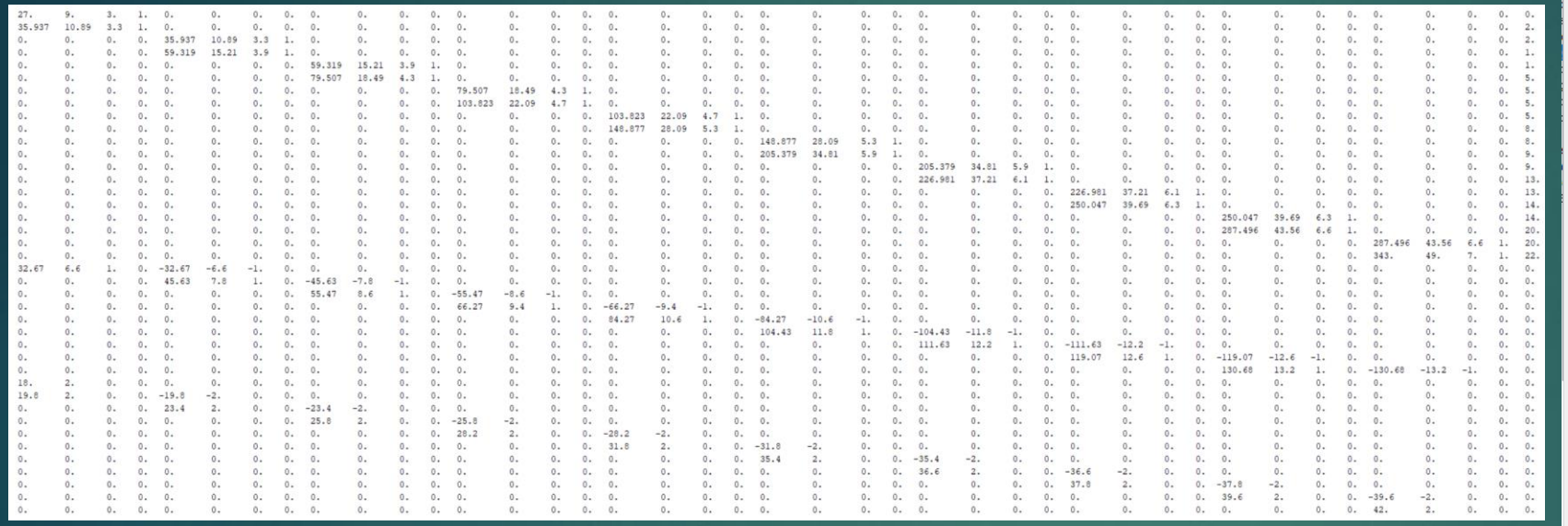
```
function Plot_SplineNatural(x,y)
    [a,b,c,d]=SplineNatural(x,y)
    N=length(x)-1 // # de intervalos
    h  =diff(x);
    dy =diff(y)./h;
    dl=h(2:N-1);
    dp=[2*(h(1:N-1)+h(2:N))];
    du=h(2:N-1);
    r=3*diff(dy)
    printf("[x h y dy r]");
    disp([x',[diff(x) %nan]',y',[diff(y)./diff(x) %nan]',...
          [%nan r %nan ]',[%nan dp %nan]' ]);
    printf("Sistema Tridiagonal [dl dp du | r = c]");
    disp([diag(dp)+diag(du,1)+diag(dl,-1) r' c(2:N)]);
    printf("Matriz de Soluções [a b c d]");
    disp([a(1:N),b,c(1:N),d])
    printf("Splines Natural\n");
    for(i=1:N)
     printf("%.3f+(%.3f)(x-%.3f)+(%.3f)(x-%.3f)^2+(%.3f)(x-%.3f)^3\n",...
            a(i),b(i),x(i),c(i),x(i),d(i),x(i))
    end
    xp=linspace(min(x),max(x),10000);
    plot(xp,InterSplineNatural(xp,x,a,b,c,d));
    scatter(x,y);
    xtitle("Interpolação Splines "+string(N)+" intervalos");
endfunction
```

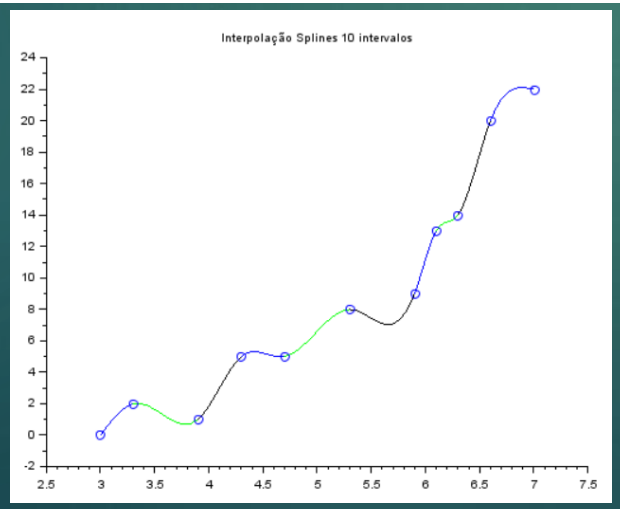
x_i	y_i
1	2
2	5
4	7
7	6
11	1




```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> Plot_SplineNatural(x,y)
[x h y dy r]
   1.    1.    2.    3.           Nan     Nan
   2.    2.    5.    1.          -6.      6.
   4.    3.    7.   -0.3333333   -4.      10.
   7.    4.    6.   -1.25        -2.75    14.
   11.   Nan   1.    Nan          Nan     Nan
Sistema Tridiagonal [dl dp du | r = c]
   6.    2.    0.    -6.     -0.9458904
   2.    10.   3.    -4.     -0.1623288
   0.    3.    14.   -2.75   -0.1616438
Matriz de Soluções [a b c d]
   2.    3.3152968    0.          -0.3152968
   5.    2.3694064   -0.9458904    0.1305936
   7.    0.152968    -0.1623288    0.0000761
   6.   -0.8189498   -0.1616438    0.0134703
Splines Cúbicas
2.000+(3.315)(x-1.000)+(0.000)(x-1.000)^2+(-0.315)(x-1.000)^3
5.000+(2.369)(x-2.000)+(-0.946)(x-2.000)^2+(0.131)(x-2.000)^3
7.000+(0.153)(x-4.000)+(-0.162)(x-4.000)^2+(0.000)(x-4.000)^3
6.000+(-0.819)(x-7.000)+(-0.162)(x-7.000)^2+(0.013)(x-7.000)^3
```

- Com 11 pontos de controle (N=10), o cálculo direto dos coeficientes das splines cúbicas gera um sistema esparso de 40 equações simultâneas (4Nx4N), o que pode gerar erros de arredondamento.
- Com o cálculo alternativo, precisamos resolver apenas um sistema tridiagonal de 9 equações simultâneas (N-1 x N-1) , o que é bem mais simples de ser resolvido e é mais estável numericamente.






```
--> x=[3,3.3,3.9,4.3,4.7,5.3,5.9,6.1,6.3,6.6,7];

--> y=[0,2,1,5,5,8,9,13,14,20,22];

--> Plot_SplineNatural(x,y)
[x h y dy r]
   3.    0.3   0.    6.6666667   Nan   Nan
   3.3   0.6   2.   -1.6666667  -25.   1.8
   3.9   0.4   1.    10.         35.   2.
   4.3   0.4   5.    0.         -30.   1.6
   4.7   0.6   5.    5.          15.   2.
   5.3   0.6   8.    1.6666667  -10.   2.4
   5.9   0.2   9.    20.         55.   1.6
   6.1   0.2   13.   5.         -45.   0.8
   6.3   0.3   14.   20.         45.   1.
   6.6   0.4   20.   5.         -45.   1.4
   7.    Nan   22.   Nan         Nan   Nan
Sistema Tridiagonal [dl dp du | r = c]
   1.8   0.6   0.    0.    0.    0.    0.    0.    0.   -25.   -24.262098
   0.6   2.    0.4   0.    0.    0.    0.    0.    0.    35.    31.119629
   0.    0.4   1.6   0.4   0.    0.    0.    0.    0.   -30.   -31.704996
   0.    0.    0.4   2.    0.6   0.    0.    0.    0.    15.    20.700356
   0.    0.    0.    0.6   2.4   0.6   0.    0.    0.   -10.   -22.864521
   0.    0.    0.    0.    0.6   1.6   0.2   0.    0.    55.    54.091063
   0.    0.    0.    0.    0.    0.2   0.8   0.2   0.   -45.   -89.134938
   0.    0.    0.    0.    0.    0.    0.2   1.    0.3   45.    77.448689
   0.    0.    0.    0.    0.    0.    0.    0.3   1.4  -45.   -48.739005
Matriz de Soluções [a b c d]
   0.    9.0928765   0.          -26.957887
   2.    1.814247   -24.262098    30.767626
   1.    5.9287651   31.119629   -52.353854
   5.    5.6946182  -31.704996    43.671126
   5.    1.292762    20.700356   -24.202709
   8.   -0.0057374  -22.864521    42.753102
   9.    18.730188   54.091063   -238.71
   13.   11.721412  -89.134938    277.63938
   14.   9.3841627   77.448689   -140.20855
   20.   17.997068  -48.739005    40.615837
Splines Cúbicas
0.000+(9.093)(x-3.000)+(0.000)(x-3.000)^2+(-26.958)(x-3.000)^3
2.000+(1.814)(x-3.300)+(-24.262)(x-3.300)^2+(30.768)(x-3.300)^3
1.000+(5.929)(x-3.900)+(31.120)(x-3.900)^2+(-52.354)(x-3.900)^3
5.000+(5.695)(x-4.300)+(-31.705)(x-4.300)^2+(43.671)(x-4.300)^3
5.000+(1.293)(x-4.700)+(20.700)(x-4.700)^2+(-24.203)(x-4.700)^3
8.000+(-0.006)(x-5.300)+(-22.865)(x-5.300)^2+(42.753)(x-5.300)^3
9.000+(18.730)(x-5.900)+(54.091)(x-5.900)^2+(-238.710)(x-5.900)^3
13.000+(11.721)(x-6.100)+(-89.135)(x-6.100)^2+(277.639)(x-6.100)^3
14.000+(9.384)(x-6.300)+(77.449)(x-6.300)^2+(-140.209)(x-6.300)^3
20.000+(17.997)(x-6.600)+(-48.739)(x-6.600)^2+(40.616)(x-6.600)^3
```

Conteúdo Opcional

Dedução do Cálculo Alternativo para Splines Cúbicas)

n=5 pontos
N=4 intervalos

x_i	y_i
1	2
2	5
4	7
7	6
11	1

Com N+1 pontos de controle $x(1:N+1)\ e\ y(1:N+1)$, podemos definir N polinômios cúbicos interpoladores, um para cada intervalo $[x_i, x_{i+1}]$ com sendo:

$$p_i(x) = a_i + b_i(x - x_i) + c_i\ (x - x_i)^2 + d_i\ (x - x_i)^3 \quad 1 < i < N \qquad (I)$$

Como o polinômio $p_i(x)$ deve passar pelo ponto de controle $[x_i, y_i]$ no início do intervalo teremos:

$$p_i(x_i) = \boxed{a_i = y_i}$$

O polinômio $p_i(x)$ passa também pelo ponto de controle $[x_{i+1}, y_{i+1}]$ no final do intervalo. No entanto, o polinômio $p_{i+1}(x)$ passar por este mesmo ponto de controle $[x_{i+1}, y_{i+1}]$, porém no início do seu intervalo

$$p_{i+1}(x_{i+1}) = y_{i+1} = p_i(x_{i+1}) = y_i + b_i(x_{i+1} - x_i) + c_i\ (x_{i+1} - x_i)^2 + d_i\ (x_{i+1} - x_i)^3$$

Se $h_i = x_{i+1} - x_i$

Temos : $y_{i+1} = y_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \qquad (II)$

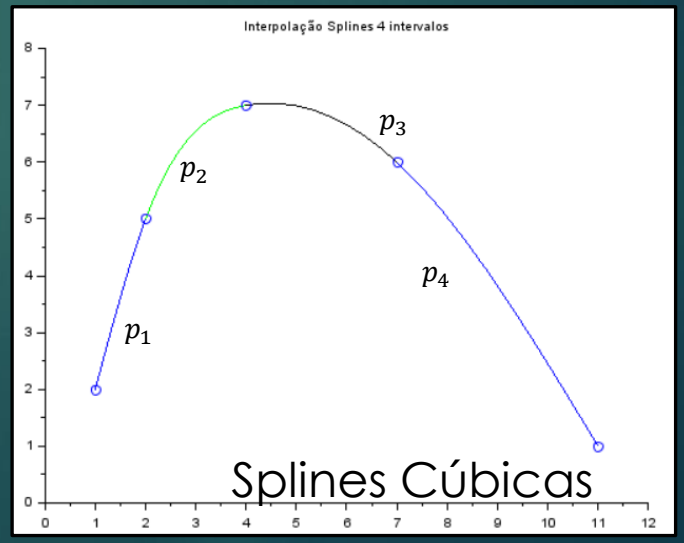
As primeiras derivadas nos pontos interiores devem ser iguais

$$p'_i(x) = b_i + 2c_i(x - x_i) + 3d_i\ (x - x_i)^2$$

$$p'_i(x_i) = b_i$$

$$p'_{i+1}(x_{i+1}) = b_{i+1} = p'_i(x_{i+1}) = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 \qquad (III)$$



As segundas derivadas nos pontos interiores devem ser iguais

$$p''_i(x) = 2c_i + 6d_i(x - x_i) \qquad (IV)$$

$$p''_i(x_i) = 2c_i$$

$$p''_{i+1}(x_{i+1}) = 2c_{i+1} = p''_i(x_{i+1}) = 2c_i + 6d_i(x_{i+1} - x_i)$$

$$c_{i+1} = c_i + 3d_i h_i \rightarrow \boxed{d_i = \frac{(c_{i+1} - c_i)}{3h_i} \qquad (V)}$$

Substituindo *(V)* em *(II)*

$$y_{i+1} = y_i + b_i h_i + c_i h_i^2 + \left[\frac{(c_{i+1} - c_i)}{3h_i}\right] h_i^3 = y_i + b_i h_i + \frac{(c_{i+1} + 2c_i)}{3} h_i^2 \rightarrow \qquad y_{i+1} = y_i + b_i h_i + \frac{(c_{i+1} + 2c_i)}{3} h_i^2 \quad (VI)$$

Substituindo *(V)* em *(III)*

$$b_{i+1} = b_i + 2c_i h_i + 3\left[\frac{(c_{i+1} - c_i)}{3h_i}\right] h_i^2 = b_i + (c_{i+1} + c_i)h_i \rightarrow \qquad b_{i+1} = b_i + (c_{i+1} + c_i)h_i \qquad (VII)$$

Se $dy_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{h_i}$

E isolando b_i em *(VI) teremos:*

$$\boxed{b_i = dy_i - \frac{h_i}{3}(c_{i+1} + 2c_i) \qquad (VIII)}$$

e subtraindo 1 ao índice 'i' de (VIII) teremos:

$$b_{i-1} = dy_{i-1} - \frac{h_{i-1}}{3}(c_i + 2c_{i-1}) \qquad (IX)$$

Subtraindo 1 do índice "i" de *(VII)* teremos

$$b_i = b_{i-1} + (c_i + c_{i-1})h_{i-1} \qquad (X)$$

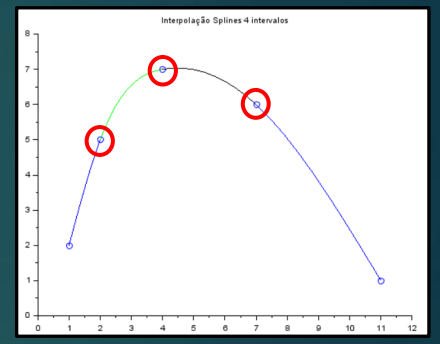
Substituindo (VIII) e (IX) em (X) teremos

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3(dy_i - dy_{i-1}) \qquad 2 < i < N \qquad (XI)$$

Podemos iniciar a equação de recorrência (XI) com $c_1 = 0$ (segunda derivada nula nos pontos externos). Lembrar que também $c_{N+1} = 0$

Esta equação de recorrência pode ser então aplicada aos ponto internos (i=2:N), o que forma um sistema tridiagonal que, uma vez resolvido, nos fornecerá c(2:N).

Acrescentando os pontos $c_1 = c_{N+1} = 0$, teremos o vetor completo c(1:N+1)



$$r_i = 3(dy_i - dy_{i-1}) \qquad (2 \le i \le N, \text{pontos internos})$$

$$h_i = x_{i+1} - x_i \qquad (1 \le i \le N)$$

i=2	$2*(h_1+h_2)$	h_2			c_2		r_2
i=3	h_2	$2*(h_2+h_3)$	h_3	*	c_3	=	r_3
i=4		h_3	$2*(h_3+h_4)$		c_4		r_4

Exemplo 5 pontos de controle N=4

Uma vez resolvido o sistema tridiagonal e calculados os coeficientes c_i, podemos utilizar as equações (V) e (VIII) para calcular os coeficientes d_i e b_i, e escrever o polinômio $p_i(x)$ da equação (I) (Lembrar que $a_i = y_i$).

$$p_i(x) = a_i + b_i(x - x_i) + c_i\,(x - x_i)^2 + d_i\,(x - x_i)^3$$

x_i	y_i
1	2
2	5
4	7
7	6
11	1

i	x_i	$h=x_{i+1}-x_i$	y_i	$dy=(y_{i+1}-y_i)/(x_{i+1}-xi)$	$r=3(dy_i-dy_{i-1})$	$2*(h_i+h_{i-1})$
1	1.00	1.00	2.00	3.00		
2	2.00	2.00	5.00	1.00	-6.00	6.00
3	4.00	3.00	7.00	-0.33	-4.00	10.00
4	7.00	4.00	6.00	-1.25	-2.75	14.00
5	11.00		1.00			

i=2	$2*(h_1+h_2)$	h_2			c_2		r_2
i=3	h_2	$2*(h_2+h_3)$	h_3	*	c_3	=	r_3
i=4		h_3	$2*(h_3+h_4)$		c_4		r_4
	6	2			c_2		-6
	2	10	3	*	c_3	=	-4
		3	14		c_4		-2.75

Resolver Sistema

-0.946
-0.162
-0.162

i	$a_i=y_i$	bi $=dy_i-h_i/3*(c_{i+1}+2c_i)$	c_i	$d_i=(c_{i+1}-ci)/(3*h_i)$
1	2.00	3.31530	0.00000	-0.31530
2	5.00	2.36941	-0.94589	0.13059
3	7.00	0.15297	-0.16233	0.00008
4	6.00	-0.81895	-0.16164	0.01347

$$p_i(x) = a_i + \mathrm{b_i}(x - x_i) + \mathrm{c_i}\ (x - x_i)^2 + \mathrm{d_i}\ (x - x_i)^3$$

- Os coeficientes a_i não são incógnitas, pois $a_i = y_i$, isto é, são os valor dos polinômios cúbicos no início do intervalo.
- O coeficientes c_i são as únicas incógnitas do sistema, e representam a metade das segundas derivadas do polinômio cúbico no início do intervalo, e são obtidos pela resolução do sistema tridiagonal.
- Os coeficientes b_i não são incógnitas e representam o valor do da primeira derivada do polinômio cúbico no início do intervalo, e podem ser avaliados pela equação $b_i = dy_i - \frac{h_i}{3}(c_{i+1} + 2c_i)$
- Os coeficientes d_i não são incógnitas, pois podem ser obtidos pela equação $\mathrm{d_i} = \frac{(\mathrm{c_{i+1}} - \mathrm{c_i})}{3h_i}$

Spline Clamped – Controle das 1^{as} derivadas no início (dy_0) e no final (dy_n)

```
function [a,b,c,d,c]=SplineClamped(x,y,dy_0,dy_n)
    N=length(x)-1 // # de intervalos
    h=diff(x);
    dy=diff(y)./h;
    dl=h(2:N-1);
    dp=2*(h(1:N-1)+h(2:N));
    r=3*diff(dy);
    du=h(2:N-1);
    dp(1)  =dp(1)  -h(1)/2;
    dp(N-1)=dp(N-1)-h(N)/2;
    r(1)  =r(1)  -3/2*(dy(1)-dy_0);
    r(N-1)=r(N-1)-3/2*(dy_n -dy(N));
    c=[0 tridiagonal(dl,dp,du,r)]'
    c(1)  =3/2*(dy(1)-dy_0) /h(1)-c(2)/2;
    c(N+1)=3/2*(dy_n -dy(N))/h(N)-c(N)/2;
    a=y';
    for k=1:N
        b(k)= dy(k)-h(k)*(2*c(k)+c(k+1))/3;
        d(k)= (c(k+1)-c(k))/(3*h(k));
    end
endfunction
```

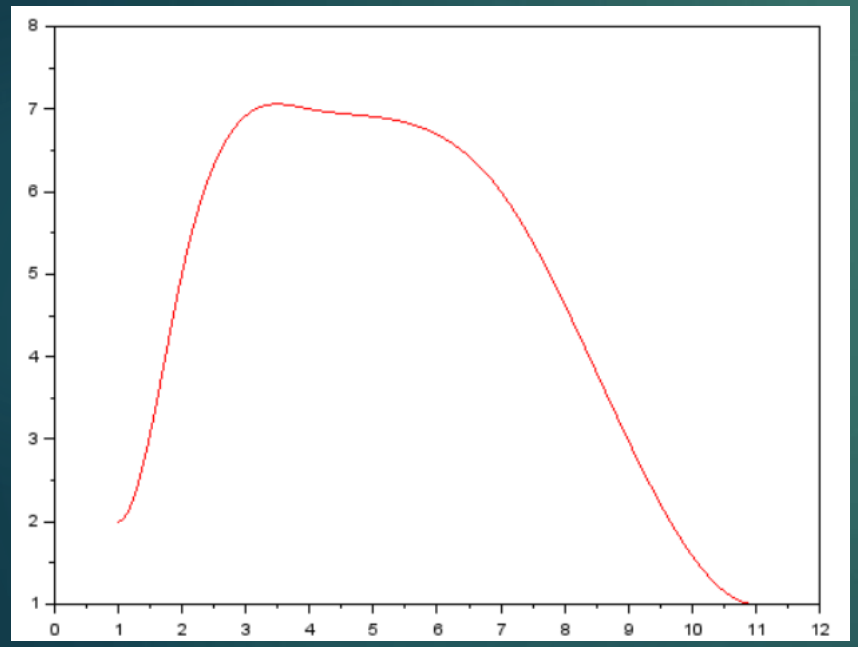
```
function [a,b,c,d]=SplineNatural(x,y)
    N=length(x)-1 // # de intervalos
    h  =diff(x);
    dy =diff(y)./h;
    dl=h(2:N-1);
    dp=[2*(h(1:N-1)+h(2:N))];
    du=h(2:N-1);
    r=3*diff(dy)
    c=[0 tridiagonal(dl,dp,du,r) 0]';
    a=y';
    for (k=1:N)
        b(k)= dy(k)-h(k)*(2*c(k)+c(k+1))/3;
        d(k)= (c(k+1)-c(k))/(3*h(k));
    end
endfunction
```

Algoritmo de campled spline:

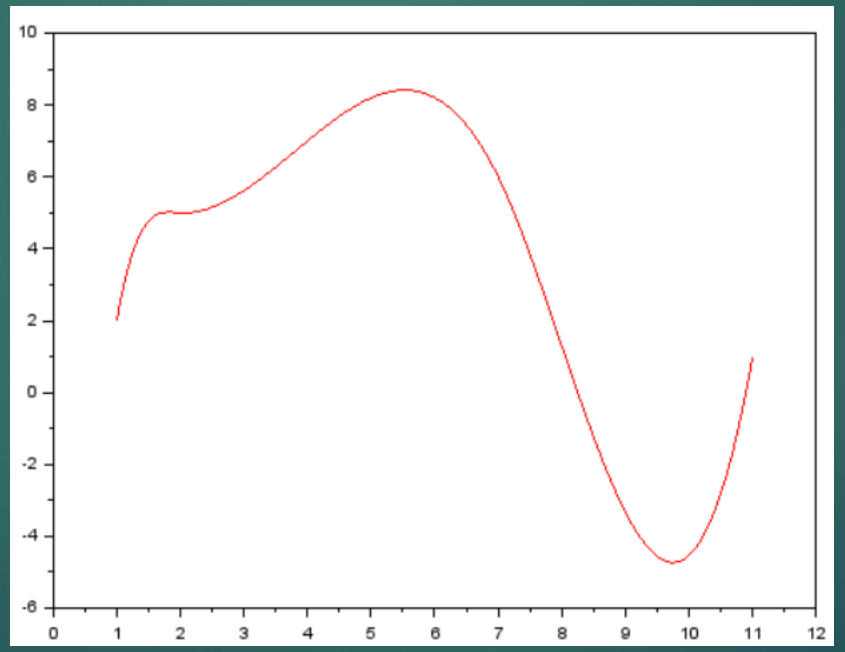
- O algoritmo clamped spline constrói uma spline que interpola os pontos fornecidos e satisfaz as condições de contorno especificadas.
- O algoritmo também envolve a resolução de um sistema linear tridiagonal para determinar os coeficientes do spline.
- O sistema linear tridiagonal é formado reforçando a continuidade da primeira e segunda derivadas em cada nó interior, semelhante ao algoritmo natural spline.
- Entretanto, no algoritmo clamped spline, condições de contorno adicionais são especificadas no primeiro e no último nós.
- As condições de contorno normalmente fixam os valores das primeiras derivadas nos dois nós exteriores, representando inclinações ou gradientes.
- A spline resultante não apenas tem um comportamento suave, mas também corresponde às derivadas especificadas nos pontos exteriores

```
--> x=[1,2,4,7,11];

--> y=[2,5,7,6,1];

--> xp=linspace(min(x),max(x),1000);
```

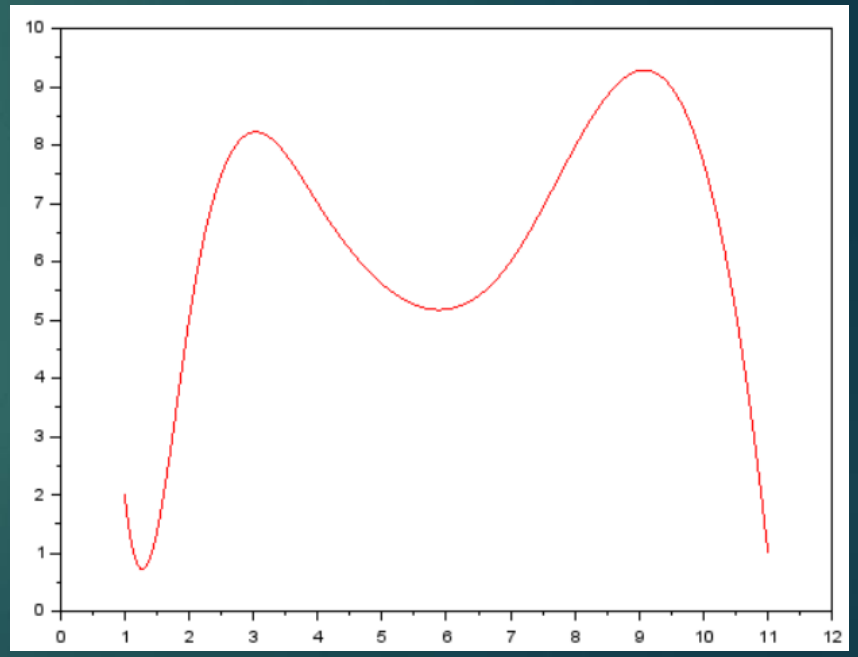
```
--> [a,b,c,d]=SplineClamped(x,y,0,0);

--> plot(xp,InterSplineNatural(xp,x,a,b,c,d),'r')
```




```
--> [a,b,c,d]=SplineClamped(x,y,10,10);

--> plot(xp,InterSplineNatural(xp,x,a,b,c,d),'r')
```




```
--> [a,b,c,d]=SplineClamped(x,y,-10,-10);

--> plot(xp,InterSplineNatural(xp,x,a,b,c,d),'r')
```



Interpolação de Shannon para pontos Equiespaçados

- Um sinal analógico $y = f(x)$ pode ser amostrado com período de amostragem T gerando a sequência de amostras $y_i = f[nT]$
- A ideia da interpolação de Shannon é reconstituir um sinal analógico $f(x)$ a partir de suas amostras $f[nT]$.
- Se um sinal amostrado é limitado em frequência, podemos reconstituí-lo através da soma de senos cardinais (sinc) . Cada amostra (intervalo amostral) contribuirá com uma função sinc.

$$y_p = f(x_p) = \sum_{i=1}^{n} y_i sinc\left(\frac{\pi}{T}(x_p - x_i)\right)$$

```
1  function yp=InterpolacaoShannon(xp,x, y)
2      n = length(x);
3      yp = xp*0;
4      T=x(2) - x(1)
5      for i = 1:n
6          yp = yp + y(i) * sinc(%pi/T*(xp-x(i)));
7      end
8      index = find(xp<min(x)|xp>max(x))
9      yp(index)=%nan
10     plot(xp,yp,'r')
11     scatter(x,y)
12 end
```