UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA - INE

Capítulo sobre

Aproximação de funções por Interpolação Polinomial

Autores: Prof. Sérgio Peters
Acad. Andréa Vergara da Silva

e-mail: sergio.peters@ufsc.br

Florianópolis, 2013.

Capítulo 5 - Teoria de Aproximações

Aproximação Polinomial por Interpolação Geral e Spline

5.0). Fundamentos:

Sabe-se que a análise numérica trata do uso de modelos não aritméticos através de quantidades aritméticas.

Nos próximos capítulos trataremos de um dos problemas centrais da Análise Numérica, que é o da aproximação de uma função y = f(x) através de outra z = g(x).

Por que aproximar y = f(x)?

Dois são os motivos básicos para a abordagem deste problema:

1º). A y = f(x) pode estar apresentada apenas na forma discreta, isto é, uma tabela do tipo que ocorre em experimentos e observações de maneira geral.

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

Ex.: Relação entre idade e produção média mensal de uma máquina (torno, robô,...).

x (meses)	1	2	3	4	5
y (10^3 peças/mês)	30	33	29	27	19

2º). A y = f(x) possui expressão conhecida, porém ineficiente ou impossível de ser utilizada.

Ex.:

(i). Construção de funções pré-definidas em bibliotecas de programas:

- e^x, sen(x), ...

(ii). Computação Gráfica:

- traçar o gráfico de $y = e^{\sqrt{\ln(x)}}$

(iii). Sistemas dedicados:

- obter valores de $y = \text{sen}\left[\ln\left(\sqrt{\text{ar coth}(x)}\right)\right]$

Neste exemplo, o tempo de CPU necessário para o cálculo da expressão acima é muito grande. Computacionalmente é mais interessante utilizar uma aproximação para se obter a resposta da expressão acima.

(iv). Manipulação Algébrica de modelos matemáticos:

- obter $\int_0^2 e^{-x^2} dx$

Quem pode ser a aproximadora z = g(x)?

Em princípio a g(x) deve ser facilmente utilizável. Ela pode ser encontrada nas seguintes famílias:

1º). Polinomiais: $g(x) = \sum_{i=0}^{n} a_i x^i = Pn(x)$

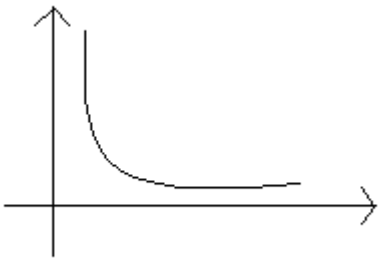
Vantagens:

- Só envolvem operações elementares;
- Facilmente deriváveis, integráveis, etc,...

- Formam um anel, ou seja, todas as transformações algébricas aplicadas resultam em um outro polinômio.

2º). Racionais: $g(x) = \frac{Pn(x)}{Qm(x)}$

Permite a representação de curvas do tipo



3º). Trigonométricas: $g(x) = \sum_{i=1}^{m} a_i \, sen(ix)$

5.1). Aproximação Polinomial pela Técnica da Interpolação:

Em virtude de suas características (simplicidade, continuidade, diferenciabilidade, etc,...) as funções polinomiais são as mais utilizadas como aproximadoras. Contudo, tais vantagens óbvias dos polinômios teriam pouca valia se não existisse prova teórica de que os mesmos podem realmente aproximar funções.

Para tal prova teórica, vamos citar o teorema de WEIRSTRAUSS:

"Seja $y = f(x)$ contínua em $[a,b]$, então $\forall \varepsilon > 0$ sempre existe $Pn(x)$ com $n(\varepsilon)$ tal que $|f(x) - Pn(x)| < \varepsilon$, $\forall x \in [a,b]$".

Para aproximar uma função $y = f(x)$ contínua em $[a,b]$ pode-se proceder como segue:

1º). Tome n+1 pontos $(x_i, y_i = f(x_i))$, com $i = 0,1,2,...,n$, desta função com $x_i \in [a,b]$:

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

com $x_0 = a$ e $x_n = b$

2º). Tome um polinômio de grau n,

$$Pn(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$ com coeficientes a_i $(i = 0,1,2,...,n)$.

Tal que o polinômio $Pn(x)$ passe por todos os pontos $(x_i, y_i = f(x_i))$, ou seja,

$Pn(x_i) = y_i \quad \Rightarrow \quad i = 0,1,2,...,n$

Daí,

$$\begin{cases} Pn(x_0) = a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = y_0 \\ Pn(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1 \\ \\ Pn(x_n) = a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = y_n \end{cases}$$

Que gera um sistema de equações lineares com n+1 incógnitas a_i e n+1 equações:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad (*)$$

cuja solução fornece os coeficientes a_i do polinômio que passa por todos os pontos $(x_i, y_i = f(x_i))$. Este polinômio é denominado de INTERPOLADOR da função y = f(x).

Note-se que os elementos u_{ij} da matriz dos coeficientes do sistema (*) U.a = y, têm lei de formação dada por:

i=0 à n $\rightarrow u_{(i+1,j)} = x_{(i)}^{j-1}$ ('i' variando de '0' a 'n' em x_i e índices da matriz 'u' variam de '1' a n+1.)

Ex: Aproxime por interpolação y = ln(x), x $\in$ [2 ; 2,15] dividindo este intervalo em n=3 (três) partes iguais e estime ln(2.14).

Tabela gerada:

i	0	1	2	3
x	2,00	2,05	2,10	2,15
y = ln(x)	0,693	0,718	0,742	0,765

Solução:

n+1 = 4 $\Rightarrow$ n = 3 $\Rightarrow$ $P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$

Aplicando o sistema (*) temos,

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 2{,}05 & (2{,}05)^2 & (2{,}05)^3 \\ 1 & 2{,}10 & (2{,}10)^2 & (2{,}10)^3 \\ 1 & 2{,}15 & (2{,}15)^2 & (2{,}15)^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0{,}693 \\ 0{,}718 \\ 0{,}742 \\ 0{,}765 \end{bmatrix}$$

Resolvendo o sistema acima por um método direto, temos:

a_0 = -1,1234
a_1 = 1,3048
a_2 = -0,1975
a_3 = -0,0004112

$$P_3(x) = -1{,}1234 + 1{,}3048x - 0{,}1975x^2 - 0{,}0004112x^3 \cong \ln(x)$$

$P_3(2{,}14) = 0{,}760\underline{48} \cong \ln(2{,}14)$ $\Rightarrow$ avaliação por Horner ou divisão sintética.

Valor Exato: ln(2,14) = 0,760805

Algoritmo Interpolador Polinomial (resolvendo um sistema de eqs. Lineares):

```
%Interpolação polinomial para grau n >= 2
a = 1;
b = 2;
n = 11;
h = (b-a)/n;
x = a : h : b;
y = log(x);
tsis = n+1;

for i = 1 : tsis
        A(i,1) = 1;
        for j = 2 : tsis
          A(i,j) = A(i,j-1)*x(i);
        endfor
        A(i, tsis+1) = y(i);
endfor
A
C = fgauss(tsis,A) % coeficientes do polinomio interpolador
np = 100;
hp = (b-a)/np;
xp = a:hp:b;
yep = log(xp);
for i = 1 : np+1
        yip(i) = resto(n, xp(i), C);
endfor
plot(x,y,"x;f(x) = ln(x);",xp,yep,"r;F(x)exata;", xp,yip,"b;Pn(x) interpolador;")

function y = resto(n, xp, C) %Calcula o valor numérico do Polinomio em um ponto xp
b(n+1) = C(n+1);
for i = n: -1 : 1
        b(i) = C(i) + xp*b(i+1);
endfor
y = b(1);
end%function
```

Fazendo uma análise minuciosa da técnica de aproximação por interpolação polinomial, pelo menos três questões fundamentais devem ser consideradas:

1ª). Será que o sistema (*) U.a = y sempre tem solução? Caso sempre exista solução, esta será única?

2ª). Será possível melhorar a eficiência computacional (menor tempo de resposta e demanda de memória) para a obtenção do Pn(x) via (*)?

3ª). Qual é o erro associado ao se tomar Pn(x) como aproximador de f(x), $\forall\ x \in [a,b]$ com $x \neq x_i$?

5.1.a). Unicidade do Interpolador:

Vamos mostrar que existe um único interpolador de grau n para toda a tabela:

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

Considere que existe um polinômio interpolador Pn(x) de grau 'n'

$Pn(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ com n+1 coeficientes a_i (i = 0,1,2,...,n) gerados a partir da solução do sistema de equações lineares abaixo, com n+1 incógnitas a_i e n+1 equações:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad (*)$$

Sabendo que o Determinante da matriz X (Matriz de Vandermonde) acima é dado por

$$Det(X) = Det\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} = \prod_{\substack{i=0,n \\ j=0;n \\ i \neq j}} (x_i - x_j)$$

$$Det(X) = (x_n - x_{n-1}).(x_n - x_{n-2}).....(x_n - x_0).(x_{n-1} - x_{n-2}).(x_{n-1} - x_{n-3}).....(x_{n-1} - x_0).....$$
$$....(x_2 - x_1).(x_2 - x_0).(x_1 - x_0).$$

Percebe-se que o Determinante de X é não nulo, sempre que os pontos x_i forem diferentes entre si, ou seja, $(x_i - x_j)$ será sempre diferente de zero para pontos x_i não repetidos e desta forma a solução do sistema (*) terá solução única.
Assim, o polinômio gerado será único para valores de x_i não repetidos, independentemente do método utilizado para a sua determinação.

5.1.b). Determinação eficiente do interpolador

Como consequência da unicidade do interpolador, pode-se tentar determiná-lo sem a geração de sistemas lineares.

Para a função tabelada abaixo:

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

propõe-se as seguintes alternativas:

1ª). Reescrever o interpolador polinomial Pn(x) via polinômios de Lagrange:

Neste caso o interpolador Pn(x) de grau n é expresso na forma de Lagrange (combinação linear de polinômios de Lagrange):

$$Pn(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

$$Pn(x) = y_0 \frac{(x-x_1)(x-x_2)\ \ldots\ (x-x_n)}{(x_0-x_1)(x_0-x_2)\ \ldots\ (x_0-x_n)} +$$
$$+ y_1 \frac{(x-x_0)(x-x_2)\ \ldots\ (x-x_n)}{(x_1-x_0)(x_1-x_2)\ \ldots\ (x_1-x_n)} +$$
$$+ \quad \cdots$$
$$+ y_n \frac{(x-x_0)(x-x_1)\ \ldots\ (x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ \ldots\ (x_n-x_{n-1})} +$$

Note que, $\begin{cases} Pn(x_0) = y_0 \\ Pn(x_1) = y_1 \\ \quad \vdots \\ Pn(x_n) = y_n \end{cases}$, portanto Pn(x) passa sobre todos os pontos tabelados.

Assim, como o interpolador Pn(x) de grau n passa sobre todos os n+1 pontos, ele será o mesmo que o obtido anteriormente, e sem a necessidade de se resolver o sistema (*) U a = y. Esta é uma consequência da unicidade do interpolador polinomial.

Ex.: Determine o interpolador de Lagrange da função:

x	0	1	3	4
y = f(x)	2	4	0	1

Estime f(2)e f(5).

Solução:

Temos n + 1 = 4 pontos $\Rightarrow$ n = 3

$$P_3(x) = y_0 \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} +$$
$$+ y_1 \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} +$$
$$+ y_2 \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} +$$
$$+ y_3 \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

$$P_3(x) = 2\frac{(x-1)(x-3)(x-4)}{(0-1)(0-3)(0-4)} + 4\frac{(x-0)(x-3)(x-4)}{(1-0)(1-3)(1-4)} +$$
$$+ 0\frac{(x-0)(x-1)(x-3)}{(3-0)(3-1)(3-4)} + 1\frac{(x-0)(x-1)(x-3)}{(4-0)(4-1)(4-3)}$$

$f(2) \cong P_3(2) = 2{,}167$

$f(5) \cong P_3(5) = 8{,}667$

Algoritmo do interpolador de Lagrange:

```
%Interpolação polinomial de Lagrange
a = 1;
```

```
b = 2;
n = 7; %7 determinado atraves do erro exato
h = (b-a)/n;
x = a : h : b;
y = log(x);
np = 100; %N. de pontos para plotar os resultados
hp = (b-a)/np;
xp = a:hp:b;
yep = log(xp);
for i = 1 : np+1
        yip(i) = lagrange(xp(i), n, x, y);
endfor
plot(x,y,"x;f(x) = ln(x);",xp,yep,"r;F(x)exata;", xp,yip,"b;Pn(x) interpolador;")


function f = lagrange(xp, n, x, y)
f = 0;
for i=1:n+1
        prod = 1;
        for j=1:n+1
          if (j != i) prod=prod*(xp-x(j))/(x(i)-x(j));endif
        endfor
        f=f+ y(i)*prod;
endfor
endfunction
```

2[a]). Interpolador de Gregory-Newton com diferenças:

Definição 1: Para a função tabelada

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

defini-se Diferença Dividida (Δy_i), no sentido ascendente, por:

$$\Delta y_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad \Rightarrow \quad \text{diferença de primeira ordem.}$$

$$\Delta^2 y_i = \Delta(\Delta y_i) = \frac{\Delta y_{i+1} - \Delta y_i}{x_{i+2} - x_i} \qquad \Rightarrow \quad \text{diferença de segunda ordem.}$$

$$\Delta^k y_i = \frac{\Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i}{x_{i+k} - x_i} \qquad \Rightarrow \quad \text{diferença de k-ésima ordem.}$$

Ex.: Determine todas as diferenças divididas da função

i	0	1	2	3
x	1	3	4	7
y = f(x)	2	0	1	3

aplicando a definição 1.

I	x_i	y_i	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$
0	1	2	-1	2/3	-1/8
1	3	0	1	-1/12	-
2	4	1	2/3	-	-
3	7	3	-	-	-

$$\Rightarrow \frac{[(-1/12)-(2/3)]}{(7-1)} = -\frac{1}{8}$$

Definição 2: Para a função tabelada

x_i	x_0	x_1	...	x_n
$y_i = f(x_i)$	y_0	y_1	...	y_n

o polinômio de grau n expresso na forma de Gregory-Newton pode ser definido como segue:

$$Pn(x) = y_0 + \sum_{k=1}^{n} \Delta^k y_0 \left[\prod_{j=0}^{k-1} (x - x_j) \right]$$

$$Pn(x) = y_0 + \Delta y_0 (x - x_0) + \Delta^2 y_0 (x - x_0)(x - x_1) + \ldots + \Delta^n y_0 (x - x_0)\ldots(x - x_{n-1})$$

Analisando os pontos tabelados temos que:

$$\begin{cases} Pn(x_0) = y_0 \\ Pn(x_1) = y_0 + \dfrac{(y_1 - y_0)}{(x_1 - x_0)}(x_1 - x_0) = y_1 \quad \text{(demais termos se anulam, pois } (x - x_1) = 0, \text{ para } x = x_1) \\ Pn(x_2) = y_2 \\ \quad \vdots \\ Pn(x_n) = y_n \end{cases}$$

Logo, Pn(x) é o único interpolador de grau n que passa sobre todos os pontos tabelados.

Ex.: Determine o interpolador para

x	1	3	4	7
y = f(x)	2	0	1	3

Estime f(5) e f(7,5).

Solução:

Temos $n+1 = 4 \Rightarrow n = 3$

$$P_3(x) = y_0 + \Delta y_0 (x - x_0) + \Delta^2 y_0 (x - x_0)(x - x_1) + \Delta^3 y_0 (x - x_0)(x - x_1)(x - x_2)$$

Usando as diferenças divididas obtidas no exemplo anterior, temos:

$$P_3(x) = 2 + (-1)(x - 1) + (2/3)(x - 1)(x - 3) + (-1/8)(x - 1)(x - 3)(x - 4)$$

$f(5) \cong P_3(5) = 2{,}233$

$f(7,5) \cong P_3(7,5) = 2,203$

Obs.: O armazenamento das diferenças divididas pode ser feito na forma de um vetor, haja vista que apenas as diferenças no ponto i = 0 são utilizadas na forma final, portanto não se deve utilizar uma matriz para este fim.

Algoritmo de Gregory-Newton:

```
%Interpolação polinomial de Gregory Newton
a = 1;
b = 2;
n = 7; %7 determinado atraves do erro exato
h = (b-a)/n;
x = a : h : b;
y = log(x);
tsis = n+1;
%diferencas divididas
for i = 1:n
        difdiv(i,1)= (y(i+1)-y(i))/(x(i+1)-x(i));
endfor
for k = 2:n
        for i = 1:tsis-k
                difdiv(i,k) = (difdiv(i+1,k-1) - difdiv(i,k-1))/(x(i+k)-x(i));
        endfor
endfor
%difdiv

np = 100;
hp = (b-a)/np;
xp = a:hp:b;
yep = log(xp);
for i = 1 : np+1
        yip(i) = gregnew(xp(i), n, x, y, difdiv);
endfor

plot(x,y,"x;f(x) = ln(x);",xp,yep,"r;F(x)exata;", xp,yip,"b;Pn(x) interpolador;")


function f = gregnew(xp, n, x, y, difdiv)
f = y(1);
for k=1:n
        prod = 1;
        for j = 1:k
                prod = prod* (xp-x(j));
        endfor
f = f + difdiv(1, k)*prod;
endfor
endfunction
```

Considerações:

(i). As estimativas das diferenças divididas também podem ser efetuadas no sentido descendente:

$$\Delta^k y_i = \frac{\Delta^{k-1} y_i - \Delta^{k-1} y_{i-1}}{x_i - x_{i-k}} \qquad \text{conferir}$$

permitindo a utilização do interpolador de Gregory-Newton na seguinte forma alternativa:

$$Pn(x) = y_0 + \sum_{i=1}^{n} \Delta^i y_0 \left[\prod_{j=0}^{i} (x - x_j) \right] \quad \text{atualizar}$$

(ii). Devido às características das expressões algébricas, Gregory-Newton é mais eficiente quando temos que efetuar muitas estimativas em uma mesma tabela, pois o cálculo das diferenças divididas pode ser avaliado previamente e reutilizado quantas vezes forem necessárias. Já o método de Lagrange é mais eficiente quando temos que efetuar estimativas em várias tabelas com os mesmos valores da variável independente.

(iii). No método de Gregory-Newton é possível acrescentar um ponto qualquer (mesmo que a sequência de pontos fique desordenada) no final de uma tabela existente e avaliar o novo interpolador correspondente através de mais uma parcela de diferenças divididas. Por exemplo, se for necessário acrescentar o ponto (9,5) a tabela do exemplo anterior procede-se da seguinte forma:

i	x_i	y_i	Δ	Δ^2	Δ^3	Δ^4
0	1	2	-1	2/3	-1/8	3/160
1	3	0	1	-1/12	1/40	-
2	4	1	2/3	1/15	-	-
3	7	3	1	-	-	-
4	9	5	-	-	-	-

E assim, acrescenta-se apenas o termo $\Delta^4 y_0 (x - x_0)(x - x_1)(x - x_2)(x - x_3)$ ao interpolador existente:

$$P_3(x) = 2 + (-1)(x-1) + (2/3)(x-1)(x-3) + (-1/8)(x-1)(x-3)(x-4) + \\ + (3/160)(x-1)(x-3)(x-4)(x-7)$$

(iv). No método de Lagrange também é possível acrescentar um ponto qualquer a uma tabela existente, mas é necessário utilizar o método de Neville que se utiliza de relações de recorrência para avaliar o novo interpolador;

(v). Para tabelas (x_i,y_i) (i = 0,1,2,...,n) com $x_{i+1} > x_i$ e espaçamentos iguais ($x_{i+1} - x_i = h$), os cálculos das diferenças divididas $\Delta^k y_i$ podem ser simplificados. Para tanto, definem-se as chamadas diferenças finitas $\Delta^k y_i$, no caso ascendentes, por:

$$\Delta y_i = y_{i+1} - y_i \qquad \Rightarrow \qquad \text{diferença de primeira ordem.}$$

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i \qquad \Rightarrow \qquad \text{diferença de k-ésima ordem.}$$

cuja relação com as diferenças divididas é a seguinte:

$$\Delta^k y_i = \frac{\Delta^k y_i}{h^k k!}$$

gerando a seguinte fórmula simplificada:

$$Pn(x) = y_0 + \sum_{i=1}^{n} \frac{\Delta^i y_0}{h^k k!}\left[\prod_{j=0}^{i}(x - x_j)\right]$$

(vi). Nas estimativas de valores na tabela (x_i,y_i) (i=0,1,2,...,n) quando se quer avaliar f(β) com:

$\beta \in (x_0, x_n)$ tem-se uma interpolação

$\beta \notin (x_0, x_n)$ tem-se uma extrapolação

caso se queira obter o valor de β correspondente ao valor γ da função tabelada

$$f(\beta) = \gamma \quad \Rightarrow \quad \beta = ?$$

procede-se uma interpolação inversa.

Ex.: Determinar quando (t = ?) o consumo atingirá o limite de 15 KW na tabela abaixo:

t(ano)	85	89	93	95	96	...	?
Consumo(KW)	5	5,7	6,2	6,7	7		15

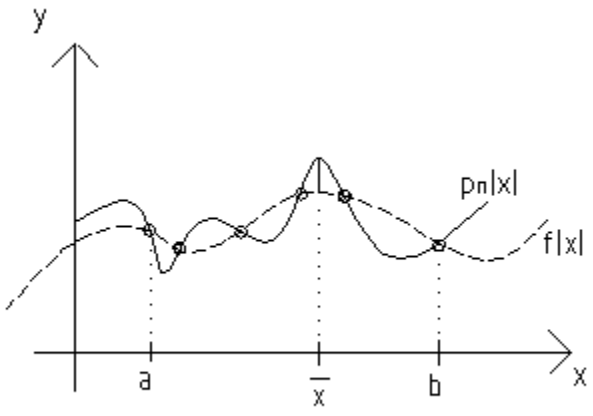
Se $\forall\ x_i\ \exists\ y_i$ distinto é só inverter a função.

(vii). O processo de interpolação polinomial também pode ser estendido a funções com várias variáveis independentes, como veremos mais adiante.

5.1.c). Avaliação do erro na interpolação polinomial

Quando se aproxima uma função y = f(x) por um interpolador polinomial Pn(x) com $x \in [x_0,x_n]$, comete-se erros de truncamento dado por:

$$E(\bar{x}) = \left|f(\bar{x}) - Pn(\bar{x})\right|,\ \forall\ \bar{x} \in [x_0, x_n] \text{ e } \bar{x} \neq x_i .$$



A delimitação do erro de truncamento $E(\bar{x})$, para qualquer $\bar{x} \in [x_0, x_n]$, indicará o grau de confiança dos resultados fornecidos por $Pn(\bar{x}) \cong f(\bar{x})$.

Teorema:

Se Pn(x) é o interpolador de

x	x_0	x_1	...	x_n
y = f(x)	y_0	y_1	...	y_n

com f(x) continuamente diferenciável em $[x_0,x_n]$, então $\forall\, \bar{x} \in [x_0, x_n]$, $\exists\, \xi \in [x_0, x_n]$ /

$$E(\bar{x}) = \left| f(\bar{x}) - Pn(\bar{x}) \right| = \left| \frac{f^{n+1}(\xi) \prod_{i=0}^{n} (\bar{x} - x_i)}{(n+1)!} \right|.$$

Note-se que a operacionalização do teorema acima é muito difícil pois:

(i). Pode ser muito dificil obter $f^{n+1}(\xi)$ (derivada n+1-ésima de f(x));
(ii). É impossível saber quem é exatamente o valor de ξ;
(iii). Para cada valor $\bar{x}$ a ser estimado tem-se que reavaliar o $E(\bar{x})$.

Tal teorema tem grande valia teórica, uma vez que:

Corolário 1 - "Sob as hipóteses do teorema anterior, se $M = \max_{x \in [x_0, x_n]} \left| f^{n+1}(x) \right|$ então

$$\left| E(\bar{x}) \right| \le \left| \frac{M}{(n+1)!} \prod_{i=0}^{n} (\bar{x} - x_i) \right| ".$$

Trata-se da majorante do erro, que é o limite superior do mesmo.

Ex.: Delimite o erro de truncamento cometido ao se aproximar f(x) = ln(x) por um interpolador polinomial em x = 0,65, considerando $x \in [0{,}4;0{,}7]$ e com n=3 subdivisões.

Solução:

Para n = 3 , então

h=(0,7-0,4)/3=0,1

i	0	1	2	3
x_i	0,4	0,5	0,6	0,7
$y_i = f(x_i)$	-0,916290732	-0,693147181	-0,510825624	-0,356674944

Aplicando o corolário 1 vem:

$$f(x) = \ln(x) \;\Rightarrow\; f'(x) = 1/x \;\Rightarrow\; f''(x) = -1/x^2 \;\Rightarrow\; f'''(x) = 2/x^3 \;\Rightarrow\; f^{iv}(x) = -6/x^4$$

Como $f^{iv}(x)$ é uma função decrescente em [0,4;0,7], o valor de M é obtido em x = 0,4

$$M = \max_{x \in [0.4;0.8]} \left| f^{iv}(x) \right| = f^{iv}(0{,}4) = 234{,}375$$

Então,

$$|E(0{,}65)| \le \left|\frac{234{,}375}{(3+1)!}\prod_{i=0}^{3}(0{,}65-x_i)\right| = \left|\frac{234{,}375}{(3+1)!}(0{,}65-x_0)(0{,}65-x_1)(0{,}65-x_2)(0{,}65-x_3)\right|$$

$$|E(0{,}65)| \le= \left|\frac{234{,}375}{(3+1)!}(0{,}65-0{,}4)(0{,}65-0{,}5)(0{,}65-0{,}6)(0{,}65-0{,}7)\right| = 9{,}15\,.10^{-4}$$

Comparando com o valor exato de ln(x) tem-se:

f(x=0,65)=ln(0,65) = -0.430782916
P_3(x=0,65) = -0.431019619 por Lagrange.
Erro exato $E(x) = |f(x) - Pn(x)|$ é 2,36703.10^{-4}.

Portanto, o erro exato 2,36703.10^{-4} é menor que o erro de truncamento máximo $|E(0{,}65)| \le 9{,}15\,.10^{-4}$.

Corolário 2 - "Se na tabela ocorrer $x_{i+1} > x_i$ (pontos ordenados) e $x_{i+1} - x_i = h$ (igualmente espaçados ∀i), então $|E(\bar{x})| \le \frac{M\,h^{n+1}}{(n+1)}$ com $M = \max_{x\in[x_0,x_n]}|f^{n+1}(x)|$ ".

Ex.: Obtenha o erro de truncamento máximo cometido ao se aproximar f(x) = e^x com $x \in [2;\ 2{,}4]$ via Pn(x) com os x_i ordenados e igualmente espaçados em n=4 intervalos. Avalie o erro exato em x=2,33.

Solução:

$$n = 4 \quad \Rightarrow \quad h = \frac{2{,}4-2}{4} = 0{,}1$$

Temos f(x) = e^x ⇒ f'(x) = e^x ⇒ f''(x) = e^x ⇒ ... ⇒ $f^{(v)}(x) = e^x$

$$M = \max_{x\in[2;2{,}4]}|f^{(v)}(x)| = |f^{(v)}(2{,}4)| = e^{2{,}4}$$

Então, $E(\bar{x}) \le \left|\frac{e^{2{,}4}(0{,}1)^5}{5}\right| = 0{,}000022$

Verificação:

P_4(2,33) = 10,2779587404
$e^{2,33}$ = 10,2779411764
Erro(real) = |10,2779587404 - 10,2779411764| = 0,000017564 < 0,000022

Ex.: Delimitar o erro máximo cometido ao se aproximar y = ln(x), x ∈[2;2.15], através do interpolador P_3(x), dividindo o intervalo [2;2.15] em 3 partes iguais.

Solução:

$$n = 3 \quad \Rightarrow \quad h = \frac{2{,}15-2}{3} = 0{,}05$$

Temos f(x) = ln(x) ⇒ f'(x) = x^{-1} ⇒ f''(x) = -1.x^{-2} ⇒ ... ⇒ $f^{(4)}(x) = 3!\ x^{-4}$ ($f^{(n+1)}(x) = n!\ x^{-(n+1)}$)

$M = \max_{x\in[2;2,15]} \left| f^{(4)}(x) \right| = 3!\ 2^{-4} = 0{,}375$

Então, $E(\bar{x}) \le \left| \frac{0{,}375.(0{,}05)^4}{4} \right| = 5{,}859.10^{-7}$

Ex.: Delimitar o erro máximo cometido ao se aproximar y = ln(x), x $\in$[1,2], através do interpolador $P_{20}(x)$, dividindo [1,2] em 20 partes iguais.

Solução:

$$n = 20 \quad \Rightarrow \quad h = \frac{2-1}{20} = 0{,}05$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \Rightarrow \ldots \Rightarrow f^{(21)}(x) = 20!\ x^{-21}$

$M = \max_{x\in[1,2]} \left| f^{(21)}(x) \right| = 20!\ (1)^{-21} = 20!$

Então, $E(\bar{x}) \le \left| \frac{20!\,(0{,}05)^{21}}{21} \right| = 5{,}5\ .\ 10^{-11}$

Ex.: Calcular o grau 'n' mínimo do interpolador polinomial $P_n(x)$ necessário para que o Erro máximo entre f(x) e o interpolador $P_n(x)$ seja menor que 1.10^{-6}, ao se aproximar f(x)=ln(x) através de $P_n(x)$, em x$\in$[1;2], dividindo o intervalo [1,2] em n partes iguais.

Solução:
Nesse caso, pode-se estabelecer valores de 'n' (INTEIROS) para calcular o valor do Erro máximo de truncamento, facilitando a obtenção das derivadas $f^{(n+1)}(x)$, pois os 'n' são números inteiros:

1a. tentativa: n=3

$$n = 3 \quad \Rightarrow \quad h = \frac{2-1}{3} = 0{,}333333333...$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \Rightarrow \ldots \Rightarrow f^{(4)}(x) = 3!\ x^{-4}$ $(f^{(n+1)}(x) = n!\ x^{-(n+1)})$

$M = \max_{x\in[1,2]} \left| f^{(4)}(x) \right| = 3!\ 1^{-4} = 6$

Então, $E(\bar{x}) \le \left| \frac{6.(0{,}33333333)^4}{4} \right| = 1{,}8519.10^{-2}$ precisamos aumentar o 'n', vamos tentar n=6

2a. tentativa: n=6

$$n = 6 \quad \Rightarrow \quad h = \frac{2-1}{6} = 0{,}16666666...$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \Rightarrow \ldots \Rightarrow f^{(7)}(x) = 6!\ x^{-7}$ $(f^{(n+1)}(x) = n!\ x^{-(n+1)})$

$M = \max_{x\in[1,2]} \left| f^{(7)}(x) \right| = 6!\ 1^{-7} = 720$

Então, $E(\bar{x}) \le \left| \frac{720.(0{,}1666667)^7}{7} \right| = 3{,}6743.10^{-4}$ precisamos aumentar o 'n', vamos tentar n=8

3a. tentativa: n=8

$$n = 8 \quad \Rightarrow \quad h = \frac{2-1}{8} = 0{,}125$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \Rightarrow \ldots \Rightarrow f^{(9)}(x) = 8!\, x^{-9}$ $(f^{(n+1)}(x) = n!\, x^{-(n+1)})$

$$M = \max_{x\in[1,2]} \left| f^{(9)}(x) \right| = 8!\ 1^{-9} = 40320$$

Então, $E(\bar{x}) \le \left| \frac{40320.(0{,}125)^9}{9} \right| = 3{,}3379.10^{-5}$ precisamos aumentar o 'n', vamos tentar n=10

6a. tentativa: n=10

$$n = 10 \quad \Rightarrow \quad h = \frac{2-1}{10} = 0{,}1$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \ldots \Rightarrow f^{(11)}(x) = 10!\, x^{-11}$ $(f^{(n+1)}(x) = n!\, x^{-(n+1)})$

$$M = \max_{x\in[1,2]} \left| f^{(11)}(x) \right| = 10!\ 1^{-11} = 3.628.800$$

Então, $E(\bar{x}) \le \left| \frac{3628800.(0{,}1)^{11}}{11} \right| = 3{,}2989.10^{-6}$ precisamos aumentar o 'n', vamos tentar n=11

7a. tentativa: n=11

$$n = 11 \quad \Rightarrow \quad h = \frac{2-1}{11} = 0{,}09090909\ldots$$

Temos $f(x) = \ln(x) \Rightarrow f'(x) = x^{-1} \Rightarrow f''(x) = -1.x^{-2} \ldots \Rightarrow f^{(12)}(x) = 11!\, x^{-12}$ $(f^{(n+1)}(x) = n!\, x^{-(n+1)})$

$$M = \max_{x\in[1,2]} \left| f^{(12)}(x) \right| = 11!\ 1^{-12} = 29.916.800$$

Então, $E(\bar{x}) \le \left| \frac{29916800.(0{,}0909090909)^{12}}{12} \right| = 1{,}0599.10^{-6}$ que é da ordem de 1.10^{-6}.

Logo, usaremos grau n=11 para calcular o polinômio interpolador Pn(x) representativo de f(x)=ln(x) em $x \in [1;2]$ com erro da ordem de 1.10^{-6} (os erros exatos devem ficar abaixo de 1.10^{-6})

Exercícios:

Repita o exemplo anterior para $f(x) = x^3+2$; $f(x) = \text{sen}(x)$; $f(x) = x.\ln(x)$; $f(x) = 1/(1+x)$ com $x \in [1,2]$ e calcule o n mínimo necessário para que o Erro Máximo em cada função seja menor que 1.10^{-6}.

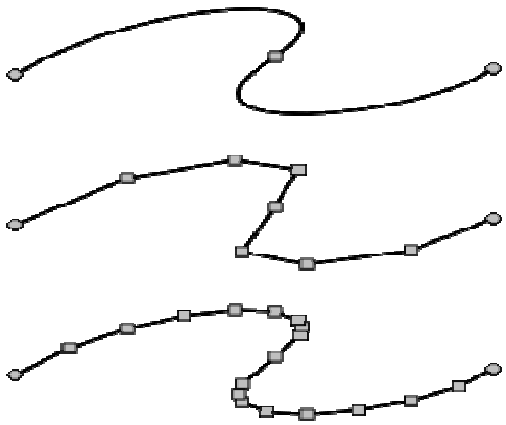
5.3 - Aproximação por Interpolação SLINE

Splines são réguas flexíveis, de madeira ou plástico, que podem ser curvadas de forma a passar por um dado conjunto de pontos (x_i,f_i) chamados nós. Foram muito utilizadas em desenhos de engenharia, nos tempos em que não se tinha recursos gráficos disponíveis para

computadores. Apesar de ser usada desde o século passado, só no fim da década de 60 foi desenvolvida a formulação matemática deste problema. Tal formalização possibilitou o desenvolvimento de vários sistemas computadorizados que utilizam aproximações gráficas de funções como CAD/CAM e TURBO GRAFIX.

Uma das maiores dificuldades encontradas ao se trabalhar com a interpolação polinomial convencional (Lagrange, Gregory-Newton, etc.) ocorre quando se toma um polinômio interpolante de grau pequeno, cometendo-se elevados erros de truncamento, enquanto que tomando-se polinômio interpolador de grau elevado, os gráficos tendem a ser altamente sinuosos (vide figura abaixo). Em certas aplicações em que a função interpolante precisa ser diferenciada, é importante obter uma aproximação tão suave quanto possível. Uma possibilidade é através de splines cúbicos, que permite interpolar inclusive expressões que não se classificam como funções, como um circulo.

Ex.:



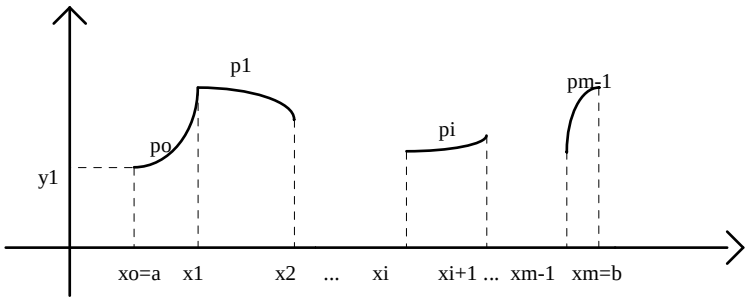
Uma técnica de aproximação possível consiste em se dividir o intervalo de interesse em vários subintervalos e interpolar separadamente, da forma mais suave possível, em cada subintervalo com polinômios de grau pequeno.

Definição 1: Interpolação Spline é uma técnica de aproximação que consiste em se dividir [a,b] de interesse e interpolar separadamente em cada intervalo com as funções mais suaves possíveis.

Definição 2: Sejam $a = x_0 < x_1 <...< x_m = b$, com 'm' subdivisões do intervalo [a,b]. Uma função spline de grau k com 'nós' nos pontos (x_i, y_i) com (i = 0,m) é uma função $p_k(x)$ com as seguintes propriedades:

a) em cada subintervalo $[x_i,x_{i+1}]$ (i=0,m-1), $p_k(x)$ é um polinômio de grau k.
b) $p_k(x)$ é contínuo em [a,b] e tem derivada contínua em [a,b] até ordem k.

A spline interpolante é a função $p_k(x)$ tal que $p_k(x_i)=f(x_i)=y_i$ (i=0,m).



5.3.1 - Spline Linear

A função interpolante spline linear de f(x) nos 'nós' x_0, x_1, ..., x_m pode ser escrita, em cada um dos 'm' subintervalos $[x_i, x_{i+1}]$ com (i=0,m-1), segundo o interpolador de Lagrange local, como

$$s_{1i}(x) = f(x_i)\frac{(x - x_{i+1})}{(x_i - x_{i+1})} + f(x_{i+1})\frac{(x - x_i)}{(x_{i+1} - x_i)}, \qquad \forall\, x \in [x_i, x_{i+1}]$$

Exemplo. Achar a spline linear em

i	0	1	2	3
x	1	2	5	7
y	1	2	3	2,5

Calculando s_{1i}, temos:

$$s_{10}(x) = f(x_0)\frac{(x - x_1)}{(x_0 - x_1)} + f(x_1)\frac{(x - x_0)}{(x_1 - x_0)} = 1\frac{(x-2)}{(1-2)} + 2\frac{(x-1)}{(2-1)} = -x + 2 + 2x - 2 = x, \quad x \in [1,2]$$

$$s_{11}(x) = 2\frac{(x-5)}{(2-5)} + 3\frac{(x-2)}{(5-2)} = \frac{1}{3}(x+4), \qquad x \in [2,5]$$

$$s_{12}(x) = 3\frac{(x-7)}{(5-7)} + 2{,}5\frac{(x-5)}{(7-5)} = \frac{1}{4}(-x+17), \qquad x \in [5,7]$$

Para obter f(3) utiliza-se a spline válida no intervalo $x \in [2,5]$, portanto calcula-se $s_{12}(3) = 1/3(3+4) = 7/3$.

Por conveniência teórica e prática, e por questões de otimização, são utilizados interpoladores polinomiais cúbicos $sp_i(x)$, com grau igual a 3 (três) fixo (Splines cúbicas). Esta técnica, e suas variantes, constituem um dos fundamentos da computação gráfica (CAD,...).

5.3.2 - Aproximação por Splines cúbicas

Para aproximar y = f(x) , x ∈ [a,b] por splines cúbicas procede-se da seguinte maneira:

1º). Dividir [a,b] em 'm' subintervalos $[x_i,x_{i+1}]$, tal que $x_{i+1} > x_i$ e $x_{i+1} - x_i = h_i$;
2º). Obter um polinômio $sp_i(x)$, de grau 3, para cada um dos 'm' subintervalos $[x_i,x_{i+1}]$,

$$sp_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \Rightarrow \quad i = 0,1,2,...m\text{-}1 \qquad (1)$$

satisfazendo as seguintes condições:

2a). $sp_i(x_i) = y_i$, para todo i = 0,1,2,...,m-1 e $sp_{m-1}(x_m) = y_m$ ($sp_i(x_i)$ passa sobre (x_i,y_i))

2b). $sp_{i-1}(x_i) = sp_i(x_i) = y_i$ para todo i = 1,2,...,m-1 ⇒ Condição de Continuidade

Note que cada polinômio $sp_i(x)$, relativo ao intervalo $[x_i,x_{i+1}]$, deve passar pelos seus dois pontos extremos (x_i,y_i) e (x_{i+1},y_{i+1}).

2c). $sp'_{i-1}(x_i) = sp'_i(x_i)$ para todo i = 1,2,...,m-1 $\Rightarrow$ Condição de Suavidade

Ou seja, em cada ponto x_i, a INCLINAÇÃO dos dois polinômios que nele incidem devem ser iguais.

2d). $sp''_{i-1}(x_i) = sp''_i(x_i)$ para todo i = 1,2,...,m-1 $\Rightarrow$ Velocidade de encurvamento

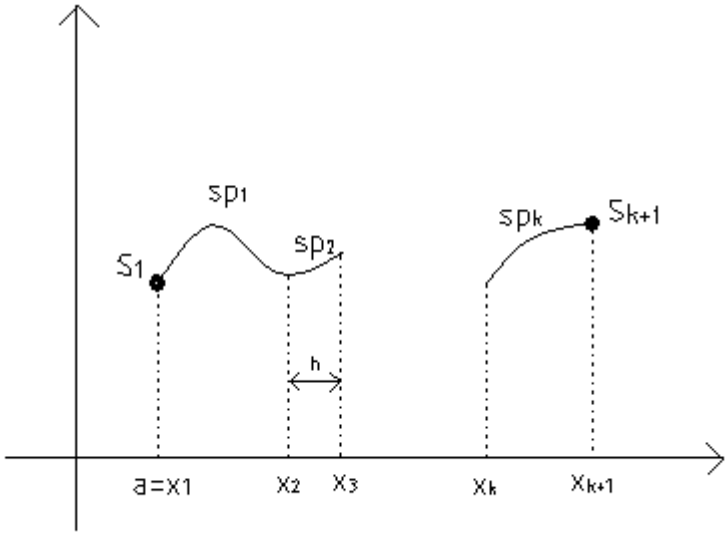
Em cada ponto, a velocidade de encurvamento (CURVATURA) dos dois polinômios que nele incidem devem ser iguais.

Obs.: Num spline mecânico estas propriedades correspondem a:

- o spline deve passar através dos 'nós'.
- o spline não quebra ou não forma ângulos agudos.
- a teoria das barras flexíveis diz que, entre os 'nós', o spline cúbico se aproxima a um polinômio cúbico.
- o spline assume a forma que minimiza a energia potencial.

3º). Para estimar f(u) com u $\in$ [a;b]:

- localizar o intervalo em que u está $\Rightarrow$ Se u $\in [x_i, x_{i+1}]$ então
- $f(u) \cong sp_i(u)$



Tomando $sp_i(x)$ para cada subintervalo $[x_i, x_{i+1}]$ com i = 0,1,2,...,m-1, através da eq.(1), e aplicando as condições estabelecidas nos itens 2a, 2b, 2c e 2d resulta que:

Aplicando a condição 2a) e 2b) na eq. (1) tem-se:

(i). $sp_i(x_i) = d_i = y_i$ (2)

$sp_i(x_{i+1}) = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = y_{i+1}$ (3)

Avaliando a primeira e segunda derivadas de $sp_i(x)$, tem-se:

$$sp_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \quad (4)$$

$$sp_i''(x) = 6a_i(x - x_i) + 2b_i \quad (5)$$

Na eq. (5) considera-se que:

$S_i = sp_i''(x_i)$ e
$S_{i+1} = sp_{i+1}''(x_{i+1}) = sp_i''(x_{i+1})$

Aplicando a condição 2d) tem-se:

(ii). $$sp_i''(x_i) = 2b_i = S_i \Rightarrow b_i = S_i / 2 \quad (6)$$

$$sp_i''(x_{i+1}) = 6a_i h_i + 2b_i = S_{i+1} \quad (7)$$

Substituindo a eq. (6) na eq. (7) tem-se

$$\Rightarrow a_i = \frac{(S_{i+1} - S_i)}{6h_i} \quad (8)$$

Substituindo as eqs. (8), (6) e (2) em (3) tem-se que

$$y_{i+1} = \frac{(S_{i+1} - S_i)h_i^3}{6h_i} + \frac{S_i h_i^2}{2} + c_i h_i + y_i \text{ então}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{S_{i+1}h_i + 2S_i h_i}{6} \quad (9)$$

Agrupando tem-se:

$$\begin{cases} a_i = (S_{i+1} - S_i) / 6h_i \\ b_i = S_i / 2 \\ c_i = (y_{i+1} - y_i) / h_i - (S_{i+1}h_i + 2S_i h_i) / 6 \\ d_i = y_i \end{cases} \quad (10)$$

Assim, tem-se os coeficientes das splines cúbicas em função dos valores de S_i e S_{i+1}. Para obter estes valores, utiliza-se a condição 2c, $sp_i'(x_i) = sp_{i-1}'(x_i)$

$sp_i'(x_i) = c_i$ e
$sp_{i-1}'(x_i) = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$

$$\Rightarrow c_i = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} \quad (11)$$

Substituindo as expressões da eq. (10) na eq. (11), tem-se

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_i S_{i+1} = 6[(y_{i+1} - y_i) / h_i - (y_i - y_{i-1}) / h_{i-1}] \quad (12)$$

para i = 1,2,3,...,m-1 (‘m-1’ equações)

De forma explícita tem-se:

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & & & & \\ & & & & & \\ & & & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{m-1} \\ S_m \end{bmatrix} = 6 \begin{bmatrix} \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ \\ . \\ . \\ \frac{y_m-y_{m-1}}{h_{m-1}} - \frac{y_{m-1}-y_{m-2}}{h_{m-2}} \end{bmatrix}$$

Como tem-se m+1 pontos com m+1 incógnitas S_i, e da eq.(12) tem-se apenas m-1 equações (i=1,2,3,...,m-1), então este é um sistema linear com m-1 equações a m+1 incógnitas. Para que se tenha solução única é necessário impor mais duas condições especiais, de preferência, nos pontos extremos de [a,b] envolvendo S_0 e S_m. Dependendo de tais condições pode-se ter vários tipos de Splines:

1) Supor que as splines cúbicas se aproximam de forma linear nos extremos, em x_0=a e x_m=b, isto é, $S_0 = S_m = 0 \Rightarrow$ **spline natural**

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & & & & \\ & & & & \\ & & h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{m-1} \end{bmatrix} = 6 \begin{bmatrix} \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ \\ . \\ . \\ \frac{y_m-y_{m-1}}{h_{m-1}} - \frac{y_{m-1}-y_{m-2}}{h_{m-2}} \end{bmatrix}$$

2) Supor que as splines aproximam-se de forma quadrática nos extremos, isto é, $S_0 = S_1$ e S_m=S_{m-1}.

$$\begin{bmatrix} (3h_0+2h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & & & & \\ & & & h_{m-2} & (2h_{m-2}+3h_{m-1}) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{m-1} \end{bmatrix} = 6 \begin{bmatrix} \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ \cdot \\ \cdot \\ \frac{y_m-y_{m-1}}{h_{m-1}} - \frac{y_{m-1}-y_{m-2}}{h_{m-2}} \end{bmatrix}$$

3) Assumir um valor para os extremos $S_0 = u$ e $S_m = v$.

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & & & & \\ & & & h_{m-2} & 2(h_{m-2}+h_{m-1}) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ \vdots \\ S_{m-1} \end{bmatrix} = 6 \begin{bmatrix} -h_0u + \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \frac{y_3-y_2}{h_2} - \frac{y_2-y_1}{h_1} \\ \cdot \\ \cdot \\ -h_{m-1}v + \frac{y_m-y_{m-1}}{h_{m-1}} - \frac{y_{m-1}-y_{m-2}}{h_{m-2}} \end{bmatrix}$$

Se for assumido um valor para os extremos a partir da extrapolação linear dos valores internos tem-se duas equações adicionais para S_0 e para S_m.

$$\frac{S_1-S_0}{h_0} = \frac{S_2-S_1}{h_1} \quad \text{e} \quad \frac{S_m-S_{m-1}}{h_{m-1}} = \frac{S_{m-1}-S_{m-2}}{h_{m-2}}$$

Que geram equações adicionais que podem ser adicionadas ao sistema dado pela eq. (12),

$$S_0 = \frac{(h_0+h_1)S_1 - h_0S_2}{h_1} \text{ e } S_m = \frac{(h_{m-2}+h_{m-1})S_{m-1} - h_{m-1}S_{m-2}}{h_{m-2}}$$

Note-se que o sistema linear resultante (com qualquer das condições apresentadas) é um **sistema tridiagonal**, que após resolvido fornece S_1, S_2, S_3,...,S_{m-1}, juntamente com a respectiva condição de extremos S_0 e S_m. Substituindo os valores de S_i nas eqs. (10) fornecem os coeficientes a_i, b_i, c_i, d_i das splines cúbicas de f(x) em [a,b].

Exemplo1. Aproximar a função tabelada abaixo por splines cúbicas, em [0,4] com h = 1, e estime f(2,5).

X	0	1	2	3	4
Y	-3	-2	5	24	61

Solução:

Então $h_i = 1$ e m = 4 intervalos.

a). Com Splines de extremos naturais (forma linear nos extremos):

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 \\ h_1 & 2(h_1+h_2) & h_2 \\ 0 & h_2 & 2(h_2+h_3) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = 6 \begin{bmatrix} [(y_2-y_1)/h_1 - (y_1-y_0)/h_0 \\ [(y_3-y_2)/h_2 - (y_2-y_1)/h_1 \\ [(y_4-y_3)/h_3 - (y_3-y_2)/h_2 \end{bmatrix}$$

Calculando os valores, obtém-se o sistema

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$ Sendo a solução S= {S_1, S_2,S_3} = { 6,42857; 10,2857; 24,4285}

com $S_0 = S_4 = 0$. Assim, para i = 0:

$$\begin{cases} a_0 = (S_1 - S_0)/6h_0 = 1{,}0714 \\ b_0 = S_0/2 = 0 \\ c_0 = (y_1 - y_0)/h_0 - (S_1 + 2S_0)/6h_0 = -0{,}0714 \\ d_0 = y_0 = -3 \end{cases}$$

$sp_0(x) = 1{,}0714(x-0)^3 + 0 - 0{,}0714(x-0) - 3$ para $x \in [0,1]$

As demais splines são as seguintes:

$sp_1(x) = 0(x-1)^3 + 0(x-1)^2 - 0,(x-0) - 2$ para $x \in [1,2]$
$sp_2(x) = 2{,}3571(x-2)^3 + 5{,}1428(x-2)^2 - 11{,}5(x-2) + 5$ para $x \in [2,3]$
$sp_3(x) = 0(x-1)^3 + 0(x-1)^2 - 0,(x-0) - 2$ para $x \in [3,4]$

Uma aproximação de f(2,5) pode ser obtida de sp_2 (2,5) = 12,33037.

b). Com Splines de extremos quadráticos:

$$\begin{bmatrix} (3h_0+2h_1) & h_1 & 0 \\ h_1 & 2(h_1+h_2) & h_2 \\ 0 & h_2 & (2h_2+3h_3) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = 6 \begin{bmatrix} [(y_2-y_1)/h_1 - (y_1-y_0)/h_0 \\ [(y_3-y_2)/h_2 - (y_2-y_1)/h_1 \\ [(y_4-y_3)/h_3 - (y_3-y_2)/h_2 \end{bmatrix}$$

Calculando os valores, obtém-se o sistema

$$\begin{bmatrix} 5 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 5 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$ Sendo a solução S = {S_1, S_2,S_3} = { 24,8; 12; 19,2}

com $S_0 = S_1 = 24{,}8$ e com $S_m = S_{m-1} = 19{,}2$. Assim,

$sp_0(x) = 2{,}4(x-0)^3 - 1{,}4(x-0) - 3$ para $x \in [0,1]$
$sp_1(x) = 0(x-1)^3 + 0(x-1)^2 - 0,(x-0) - 2$ para $x \in [1,2]$

$sp_2(x) = 1{,}2(x-2)^3 + 6(x-2)^2 + 11{,}8(x-2) + 5 \quad \text{para } x \in [2,3]$

$sp_3(x) = 0(x-1)^3 + 0(x-1)^2 - 0,(x-0) - 2 \quad \text{para } x \in [3,4]$

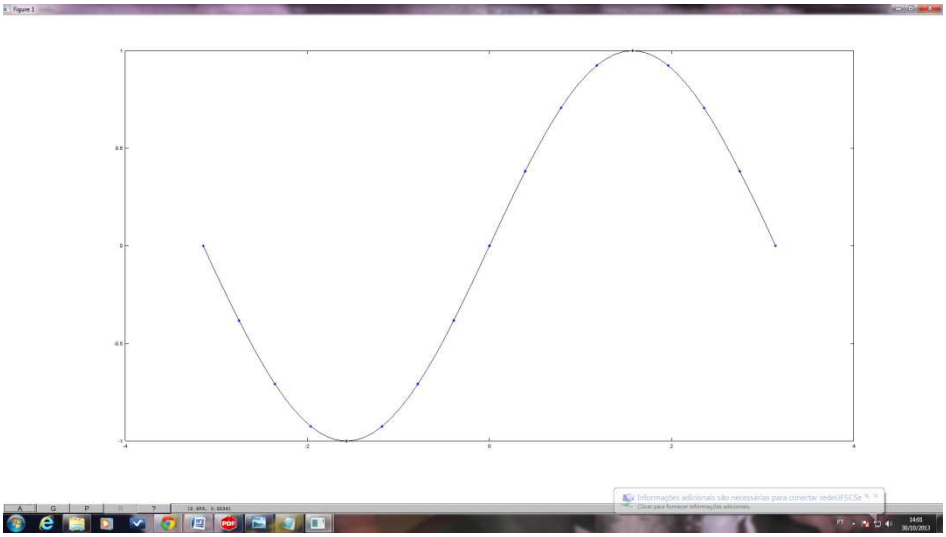
Aproximação de f(2,5) pode ser obtida por sp_2 (2,5) = 12,55.

Para efeito de comparação tem-se que f(x) tabelada corresponde a $f(x) = x^3-3$, e cujo valor exato de f(2,5) = 12,625. Portanto com spline de extremos quadráticos tem-se um resultado mais realístico.

Algoritmo Splines cúbicas:

```
% Represente a funçao sen(x) com x  entre -π e π, de forma continua e suave
% Neste caso usaremos a interpolação por splines cúbicas, que representam relações entre
% variáveis, que NÃO são funçoes.
a=-pi %valor inicial do intervalo
b=+pi %valor final do intervalo
m=16 %numero de subdivisões do intervalo [a ; b] para m+1 pontos
tx=a:(b-a)/m:b; %variavel auxiliar
x=tx
y=f1(tx) % valores exatos da função f2 = sen(tx) para os m+1 pontos tabelados
for i=1:m
        h(i)=x(i+1)-x(i);
end
h
%teremos 'm' splines cúbicas, uma para cada intervalo, formando um sistema tridiagonal de 'm-1' equacoes para S
(derivadas de segunda ordem)
t(2)=0; r(2)=3*h(1)+2*h(2); d(2)=h(2);      b(2)=6*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1)); %para S1
for i=3:m-1
 t(i)=h(i-1); r(i)=2*(h(i-1)+h(i)); d(i)=h(i); b(i)=6*((y(i+1)-y(i))/h(i)-(y(i)-y(i-1))/h(i-1)); %para Si
endfor
t(m)=h(m-1); r(m)=(2*h(m-1)+3*h(m)); d(m)=0;   b(m)=6*((y(m+1)-y(m))/h(m)-(y(m)-y(m-1))/h(m-1)); %para Sm
t
r
d
b
for i=3:m
   aux=t(i)/r(i-1);t(i)=0;
   r(i)=r(i)-aux*d(i-1);
        b(i)=b(i)-aux*b(i-1);
end
S(m)=b(m)/r(m);
for i=m-1:-1:2
  S(i)=(b(i)-d(i)*S(i+1))/r(i);
end
% Splines quadráticas NAS PONTAS da figura S1=S2 e Sm+1=Sm
% Calcula-se cada conjunto de coeficientes a, b, c, d dos polinimois de 3o. grau
S(1)=S(2); S(m+1)=S(m);
S
for i=1:m
        a(i)=(S(i+1)-S(i))/(6*h(i));
        b(i)= S(i)/2;
        c(i)=(y(i+1)-y(i))/h(i)-(S(i+1)+2*S(i))*h(i)/6;
        d(i)= y(i);
end
np=4;    %4 sub-divisões para cada sub-intervalo entre xi e xi+1
xpp=[];ypp=[];
for i=1:m
   xp=x(i):(x(i+1)-x(i))/np:x(i+1);
        for k=1:np+1
                yp(k)=a(i)*(xp(k)-x(i))*(xp(k)-x(i))*(xp(k)-x(i))+b(i)*(xp(k)-x(i))*(xp(k)-x(i))+c(i)*(xp(k)-x(i))+d(i);
        end
        xpp=[xpp xp];ypp=[ypp yp];
end
xpp
ypp
```

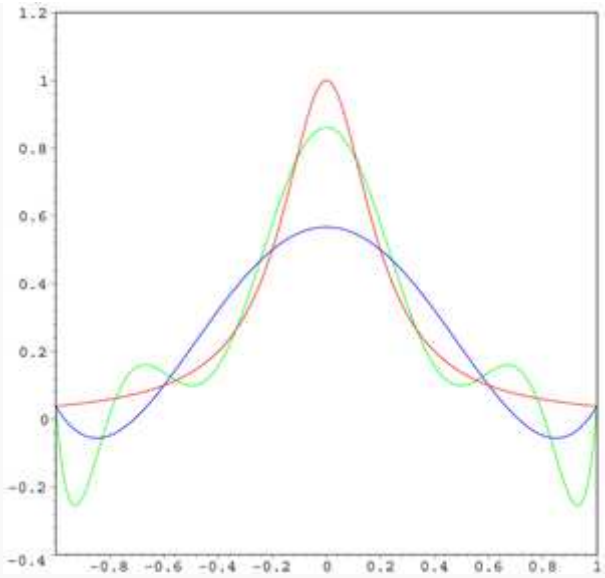
```
plot(x,y,'*',xpp,ypp,'k')

function f=f1(x)
        f=sin(x);
endfunction
```



Exemplo: função sen(x).

Algoritmo para tratamento do **Fenômeno de Runge**

Em matemática, em particular no campo específico da análise numérica, o **fenômeno de Runge** é um problema de oscilação nas bordas de um intervalo, que ocorre quando se usa interpolação polinomial com polinômios de ordem elevada. Foi descoberto por Carl Runge quando investigava erros na interpolação polinomial para aproximar certas funções



A curva vermelha é a Função EXATA de Runge.

A curva azul é uma interpolação polinomial de 5ª ordem (usando seis pontos de interpolação igualmente espaçados).

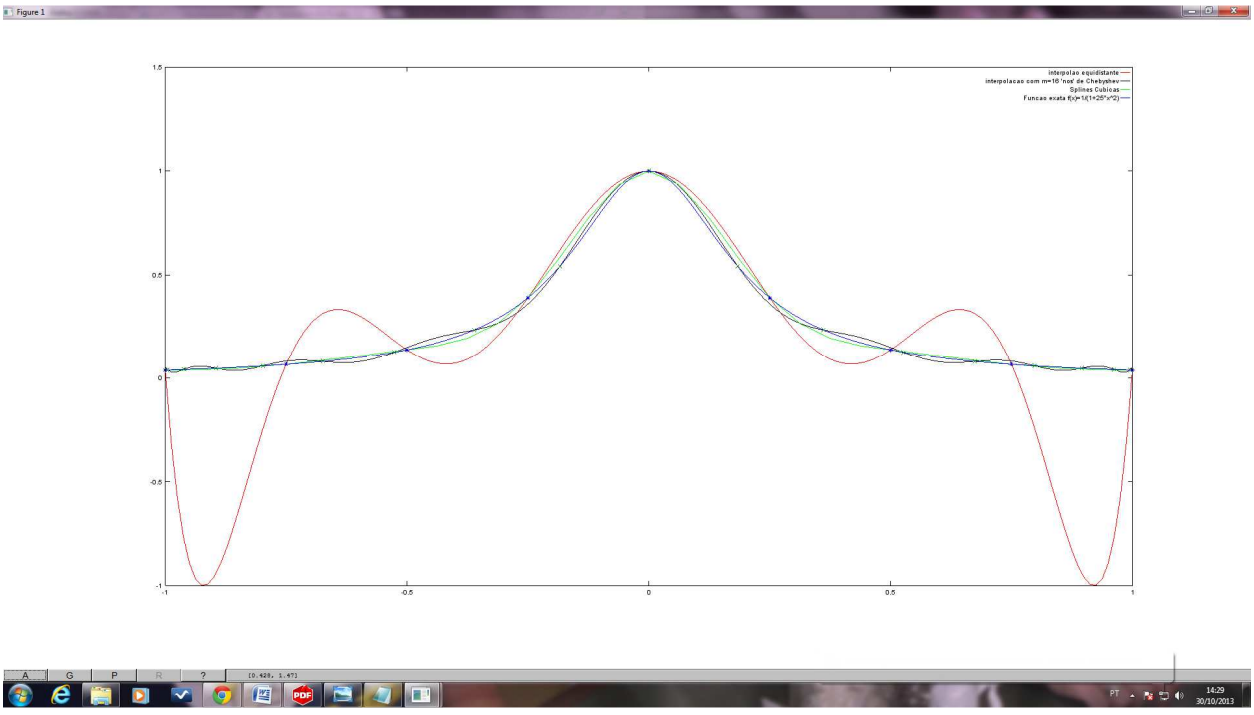
A curva verde é uma interpolação polinomial de 9ª ordem (usando dez pontos de interpolação igualmente espaçados).

A oscilação pode ser minimizada usando-se os nós de Chebyshev em vez de nós eqüidistantes para ancorar o polinômio interpolador. Neste caso, o erro máximo diminui quando a ordem do polinômio aumenta.

% Neste exemplo usaremos a interpolação:

```
%1.Polinomial com m pontos equidistantes;
%2. Polinomial com mk pontos definidos pelas raízes do polinomio de grau mk=nk+1 de Chebyshev;
%3. por splines cubicas, que representam relaçoes entre variaveis.
clear
a=-1 %valor inicial do intervalo (NÃO PODE GERAR DERIVADA NULA)
b=+1 %valor final do intervalo
n=8 %numero de subdivicoes do intervalo [a ; b] para m+1 pontos
m=n+1;


%1.Polinomial com pontos equidistantes;
x=a:((b-a)/n):b %variavel auxiliar
y=1./(1.+25.*x.*x) % valores exatos da função f(x) para os m+1 pontos tabelados
% Cálculo por interpolação polinomial
for i=1:m
  a1(i,1)=1;
  for j=2:m
    a1(i,j)=a1(i,j-1)*x(i);
  endfor
  a1(i,m+1)=y(i);
endfor
a1
coef=fgauss(m,a1)
np=20*n % numero de subdivicoes para serem plotados
xp= a:((b-a)/np):b; % (np+1) pontos x a serem plotados
for i=1:np+1
        yp(i)=Pn(n,coef,xp(i)); % (np+1) pontos y a serem plotados
        ye(i)=1/(1+25*xp(i)*xp(i));
endfor


%2. Polinomial com m pontos definidos pelas raízes do polinomio de grau m=n+1 de Chebyshev;
nk=16;mk=nk+1; % 17 nós de Chebychev, polinomio de grau nk=16
for i=1:mk
        tk(i)=cos((2*i-1)*pi/(2*mk));
        yaux(i)=1/(1+25*tk(i)*tk(i));
endfor
yk=1./(1.+25.*tk.*tk) % valores exatos da função f(t) para m ponto
% Cálculo por interpolação polinomial
for i=1:mk
  a1(i,1)=1;
  for j=2:mk
    a1(i,j)=a1(i,j-1)*tk(i);
  endfor
  a1(i,mk+1)=yk(i);
endfor
a1
coef=fgauss(mk,a1)

for i=1:np+1
        ypk(i)=Pn(nk,coef,xp(i)); % (np+1) pontos y a serem plotados
endfor



%3.  Cálculo por n Splines
for i=1:n
        h(i)=x(i+1)-x(i);
end
h
%teremos 'n' splines cubicas, uma para cada intervalo, formando um sistema tridiagonal de 'm-1' equacoes para S (derivadas de segunda ordem)
t(2)=0;      r(2)=3*h(1)+2*h(2);    d(2)=h(2);b(2)=6*((y(3)-y(2))/h(2)-(y(2)-y(1))/h(1)); %para S1
for i=3:n-1
 t(i)=h(i-1);r(i)=2*(h(i-1)+h(i));  d(i)=h(i);b(i)=6*((y(i+1)-y(i))/h(i)-(y(i)-y(i-1))/h(i-1)); %para Si
endfor
t(n)=h(n-1); r(n)=(2*h(n-1)+3*h(n));d(n)=0;   b(n)=6*((y(n+1)-y(n))/h(n)-(y(n)-y(n-1))/h(n-1)); %para Sm
t
r
d
```

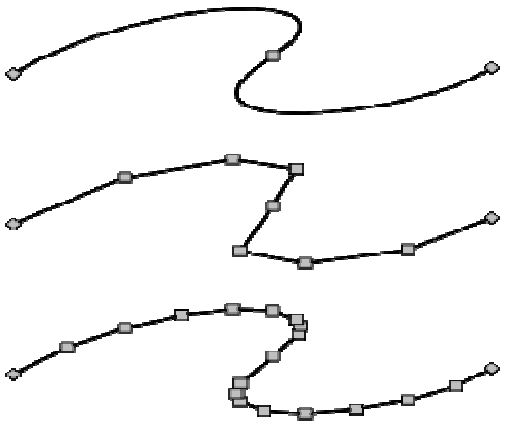
```
b
for i=3:n
   aux=t(i)/r(i-1);t(i)=0;
   r(i)=r(i)-aux*d(i-1);
            b(i)=b(i)-aux*b(i-1);
end
S(n)=b(n)/r(n);
for i=n-1:-1:2
  S(i)=(b(i)-d(i)*S(i+1))/r(i);
end
% Splines quadraticas S1=S2 e Sm+1=Sm
% Calcula-se cada conjunto de coeficientes a, b, c, d dos polinimois de 3o. grau
S(1)=S(2); S(n+1)=S(n);
S
for i=1:n
            a(i)=(S(i+1)-S(i))/(6*h(i));
            b(i)= S(i)/2;
            c(i)=(y(i+1)-y(i))/h(i)-(S(i+1)+2*S(i))*h(i)/6;;
            d(i)= y(i);
end
np=4;    %4 sub-divisões internas em cada sub-intervalo entre xi e xi+1
xpp=[];ypp=[];
for i=1:n
   xs=x(i):(x(i+1)-x(i))/np:x(i+1);
            for k=1:np+1
                     ys(k)=a(i)*(xs(k)-x(i))*(xs(k)-x(i))*(xs(k)-x(i))+b(i)*(xs(k)-x(i))*(xs(k)-x(i))+c(i)*(xs(k)-x(i))+d(i);
            end
            xpp=[xpp xs];ypp=[ypp ys];
end
xpp;
ypp;
plot(x,y,'*',xp,yp,"r;interpolação equidistante;",tk,yaux,'x',xp,ypk,"k;interpolacao com m=16 'nos' de Chebyshev;",xpp,ypp,"g;Splines Cubicas;",xp,ye,"b;Funcao exata f(x)=1/(1+25*x^2);")
%plot(x,y,'*',tk,yaux,'x',xp,ypk,"k;interpolação nos de Chebyshev;",xpp,ypp,"g;Splines Cubicas;",xp,ye,"b;Funcao exata f(x)=1/(1+25*x^2);")
```



Curvas de Bezier:

Este assunto é a base matemática de sistemas de computação gráfica, como o corel draw. Foi desenvolvido pelo engenheiro Pierre Bezier (funcionário da Renault) na década de 1960 (sistema Unisurf) para simulaçãod e layout de automóveis.

Questão preliminar: Como seguir um caminho, não funcional (com mais de um valor de y para cada x), do tipo esboçado?



Somente usando o conceito de parametrização da curva (=caminho), via:

a). tomar (n+1) pontos amostrais $p_i=(x_i,y_i)$, i=0,...,n, definidos na base de dados da curva (do caminho):

x_i	x_0	x_1	...	x_{n-1}	x_n
y_i	y_0	y_1	...	y_{n-1}	y_n

b). adotar uma nova variável independente $t\in[0;1]$, variável de controle, e dividir o intervalo [0;1] em n partes iguais de comprimento h=(1-0)/n e gerar t_i com $t_0=0$ e $t_{i+1}=t_i+h$ para i=0,... n-1.

t_i	t_0	t_1	...	t_{n-1}	t_n
p_i	p_0	p_1	...	p_{n-1}	p_n

Separando cada ponto $p_i=(x_i,y_i)$, i=0,...,n, em seus compontnes x_i e y_i

t_i	t_0	t_1	...	t_{n-1}	t_n
x_i	x_0	x_1	...	x_{n-1}	x_n

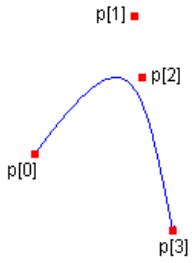
t_i	t_0	t_1	...	t_{n-1}	t_n
y_i	y_0	y_1	...	y_{n-1}	y_n

c). aproximar as duas tabelas que são funções e gerar os pontos novos do caminho usando estes dois aproximadores (interpoladores, por exemplo).

Aproximação unidimensional de Bezier:

c1). tomar (n+1) pontos referenciais do desenho desejado => $p_i=(x_i,y_i)$, i=0,...,n e expressá-los na forma parametrizada => $p_i=(x_i(t),y_i(t))$, com $t\in[0;1]$;

Por exemplo para n=3 (4 pontos).



c2). subdividir (se necessário) os (n+1) pontos referenciais e subconjuntos com (k+1) pontos, 2<=k<n;

c3). Para cada subconjunto de (k+1) pontos pi, i=0,...,k obter o seu polinômio aproximador de Bernstein $B_k(t)$de grau k:

$B_k(t) = \sum_{i=0}^{k} C_k^i . (1-t)^{k-i} . t^i . p_i$	$Bx_k(t) = \sum_{i=0}^{k} C_k^i . (1-t)^{k-i} . t^i . x_i$
	$By_k(t) = \sum_{i=0}^{k} C_k^i . (1-t)^{k-i} . t^i . y_i$

Onde $C_k^i = \frac{k!}{(k-i)!.i!}$

Forma matricial de $B_k(t)$:

$$B_k(t) = [C_k^0 . p_0 \quad C_k^1 . p_1 \quad \dots \quad C_k^k . p_k] \begin{bmatrix} (1-t)^k \\ (1-t)^{k-1} . t^1 \\ . \\ . \\ . \\ t^k \end{bmatrix}$$

Exemplos:

k=2 => Bezier quadrática.	$Bx_2(t) = 1.(1-t)^{2-0} . t^0 . x_0 + 2.(1-t)^{2-1} . t^1 . x_1 + 1.(1-t)^{2-2} . t^2 . x_2$
$B_2(t) = 1.(1-t)^{2-0} . t^0 . p_0 + 2.(1-t)^{2-1} . t^1 . p_1 + 1.(1-t)^{2-2} . t^2 . p_2$ =>	$By_2(t) = 1.(1-t)^{2-0} . t^0 . y_0 + 2.(1-t)^{2-1} . t^1 . y_1 + 1.(1-t)^{2-2} . t^2 . y_2$

k=3 => Bezier cúbica.

$B_3(t) = 1.(1-t)^{3-0} . t^0 . p_0 + 3.(1-t)^{3-1} . t^1 . p_1 + 3.(1-t)^{3-2} . t^2 . p_2 + 1.(1-t)^{3-3} . t^3 . p_3$

$B_3(t)$ =>	$Bx_3(t) = 1.(1-t)^{3-0} . t^0 . x_0 + 3.(1-t)^{3-1} . t^1 . x_1 + 3.(1-t)^{3-2} . t^2 . x_2 + 1.(1-t)^{3-3} . t^3 . x_3$ $Bx_3(t) = ax . t^3 + bx . t^2 + cx . t^1 + dx . t^0$ => dx= x_0 , cx=3.(x_1 -x_0) ,. bx=3.(x_2 -x_1)-cx , ax=(x_3 -x_0)-(cx+bx)
	$By_3(t) = 1.(1-t)^{3-0} . t^0 . y_0 + 3.(1-t)^{3-1} . t^1 . y_1 + 3.(1-t)^{3-2} . t^2 . y_2 + 1.(1-t)^{3-3} . t^3 . y_3$ $By_3(t) = ay . t^3 + by . t^2 + cy . t^1 + dy . t^0$ => dy= y_0 , cy=3.(y_1 -y_0) , by=3.(y_2 -y_1)-cy , ay=(y_3 -y_0)-(cy+by)

Essa última forma de se expresser um polinômio $B_k(t)$ é a mais adequada para implementação em computador.

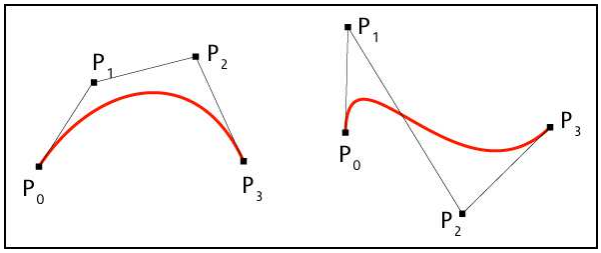
A plotagem de uma polinomial $B_k(t)$, com t∈[0;1], gera uma curva com as seguintes propriedades.

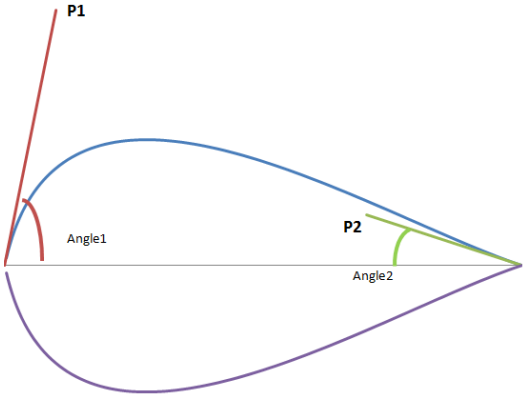
P1: " $\begin{cases} Bx_k(0) = x_0 \\ By_k(0) = y_0 \end{cases}$ e $\begin{cases} Bx_k(1) = x_k \\ By_k(1) = y_k \end{cases}$ $B_k(t)$ incia em p_0 e termina em p_k. "

P2: "A reta definida por {p_0,p_1} é a reta tangente a $B_k(t)$ em p_0, e a reta definida por {p_{k-1}, p_k} é tangente a $B_k(t)$ em p_k".

P3: "A curva gerada pela $B_k(t)$ $\{Bx_k(t), By_k(t)\}$, t∈[0;1], está contida sempre no menor polígono convexo que contém os k+1 pontos de referência da mesma."

Exemplos:

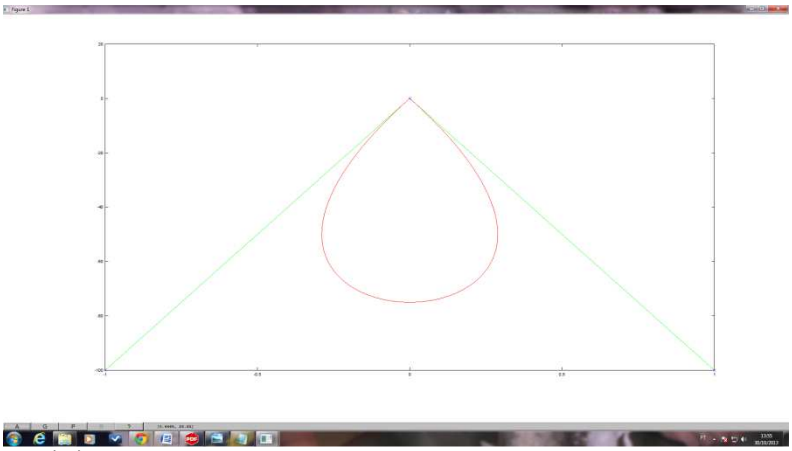




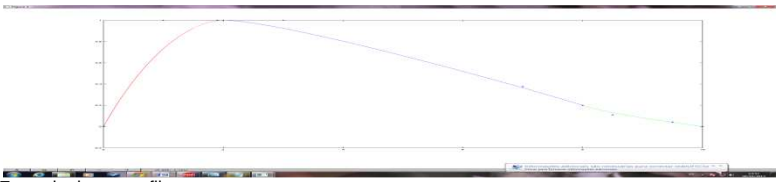
Algoritmo:

%Curvas de Bezier
clear
n=4 %pontos

```
%y=[0 0 4  4];x=[0 4 4  0]; %parabola horizontal
%y=[0 9 9  0];x=[4 4 0  0]; %parabola vertical
%y=[3 6 0 -3];x=[0 5 6  1]; %parabola inclinada
%y=[3 1 5 -3];x=[0 2 2  3]; %parabola distorcida
%y=[0 4 4  0];x=[0 5 0  5]; %cuspide
y=[0 -100 -100 0];
x=[0  1  -1  0]; %Gota
x(n+1)=x(1);y(n+1)=y(1);
n=1000
h=1/n %Espaçamento do parametro t
t=0
cx=3*(x(2)-x(1));bx=3*(x(3)-x(2))-cx;ax=(x(4)-x(1))-(cx+bx);
cy=3*(y(2)-y(1));by=3*(y(3)-y(2))-cy;ay=(y(4)-y(1))-(cy+by);
xmax=0;ymax=0;
for i=1:n+1
   xx(i)=x(1)+t*(cx+t*(bx+t*ax));
   yy(i)=y(1)+t*(cy+t*(by+t*ay));
   t=t+h;
endfor
plot(x,y,'g',x,y,'x',xx,yy,'r')
```



Exemplo de uma gota:



Exemplo de um perfil: