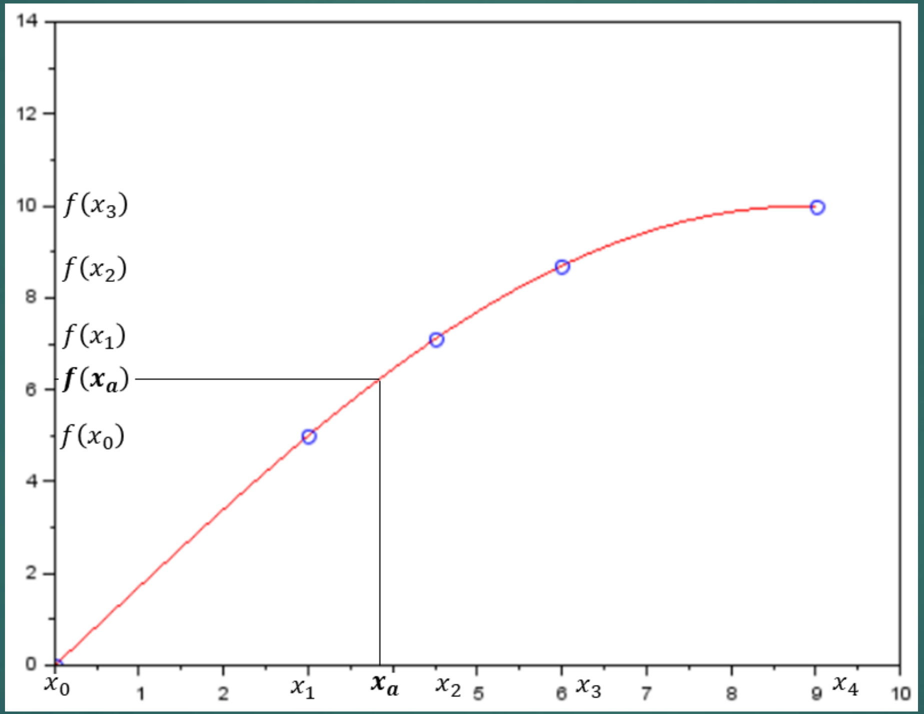
Métodos Numéricos para Engenharia

MÓDULO 10 – INTERPOLAÇÃO POR POLINÔMIOS

PROFESSOR LUCIANO NEVES DA FONSECA

Definições - Interpolação

- Interpolação é a determinação ou estimativa dos valores de uma função $f(x)$, a partir do conhecimento de apenas alguns valores da função no intervalo de interesse $x_0 < x < x_n$ (n+1 pontos de controle.
- Se forem conhecidos $n+1$ pontos de controle(x_i, y_i), com $0 < i < n$ (pontos azuis no gráfico abaixo), então o valor estimado de $f(x_a)$ é considerado uma interpolação da função no ponto x_a.



- Se $f(x)$ for um função contínua no intervalo [a,b], e conhecermos n+1 pontos de controle(medidas de laboratório, então existe um único polinômio $p_n(x)$ que passa exatamente por todos estes pontos.
- O polinômio interpolador $p_n(x)$ (linha vermelha) pode ser diretamente calculado a partir dos pontos conhecidos.

Exempo: Ajuste de um polinômio a 2 pontos de controle

- Suponha que só conheçamos o valor de uma função $f(x)$ em dois pontos de controle (segundo a tabela abaixo, círculos azuis no gráfico), provavelmente obtidos em medições no laboratório. Podemos estimar valor da função entre estes dois pontos de controle através de uma interpolação linear (linha vermelha no gráfico).

i	x	y
1	3.0	5.0
2	6.0	8.7

```
--> x=[3.0, 6.0];

--> A=[(x^0)',(x^1)']
 A  =

   1.   3.
   1.   6.

--> y=[5.0;8.7];

--> c=inv(A)*y
 c  =

   1.3000000
   1.2333333
```

Interpolação linear - polinômio de ordem 1 – $p_s(x)$

$$p_s(x) = a_0 + a_1 x$$

Como o polinômio deve passar pelos 2 pontos de controle temos:

$$p_s(3.0) = a_0 + a_1(3.0) = 5.0$$

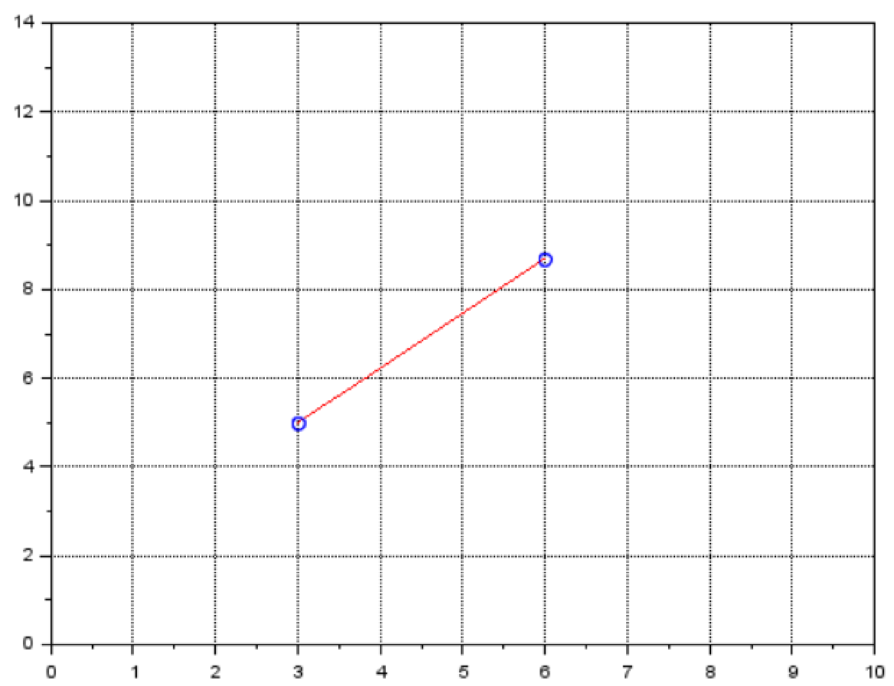
$$p_s(6.0) = a_0 + a_1(6.0) = 8.7$$

$$\begin{bmatrix} 1 & 3.0 \\ 1 & 6.0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 5.0 \\ 8.7 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1.3 \\ 1.2333 \end{bmatrix}$$

$$p_s(x) = 1.3 + 1.2333\ x$$

```
--> ps=poly(c,"s","coeff")
 ps  =

  1.3 +1.2333333s

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

--> plot(xi,horner(ps,xi),'r')

--> scatter(x,y)

--> xgrid()
```



Exempo: Ajuste de um polinômio a 3 pontos de controle

- Suponha que agora conheçamos o valor da função $f(x)$ em apenas 3 pontos de controle. Podemos interpolar um polinômio de ordem 2 a passar por estes 4 pontos.

i	x	y
0	3.0	5.00
1	4.5	7.1
2	6.0	8.7

```
--> x=[3.0, 4.5, 6.0];

--> A=[(x^0)',(x^1)',(x^2)']
 A  =

   1.   3.    9.
   1.   4.5   20.25
   1.   6.    36.

--> y=[5.0;7.1;8.7];

--> c=inv(A)*y
 c  =

  -0.7000000
   2.2333333
  -0.1111111
```

Interpolação quadrática - polinômio de ordem 2 - $P_2(x)$

$$p_s(x) = a_0 + a_1 x + a_2 x^2$$

Como o polinômio deve passar pelos 3 pontos de controle temos:

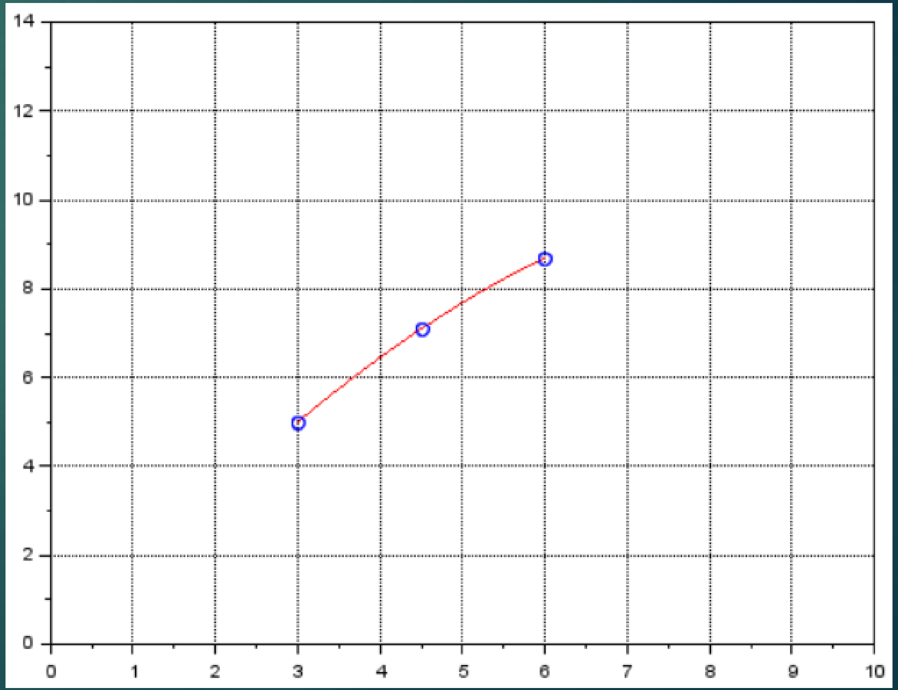
$$p_s(3.0) = a_0 + a_1(3.0) + a_2(3.0)^2 = 5.0$$
$$p_s(4.5) = a_0 + a_1(4.5) + a_2(4.5)^2 = 7.1$$
$$p_s(6.0) = a_0 + a_1(6.0) + a_2(6.0)^2 = 8.7$$

$$\begin{bmatrix} 1 & 3 & 3^2 \\ 1 & 4.5 & 4.5^2 \\ 1 & 6 & 6^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 5.0 \\ 7.1 \\ 8.7 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -0.7 \\ 2.2333 \\ -0.1111 \end{bmatrix}$$

$$p_s(x) = -0.7 + 2.2333x - 0.1111x^2$$

```
--> ps=poly(c,"s","coeff")
 ps  =

  -0.7 +2.2333333s -0.1111111s²

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

--> plot(xi,horner(ps,xi),'r')

--> scatter(x,y)

--> xgrid()
```



Exempo: Ajuste de um polinômio a 4 pontos de controle

- Suponha que agora conheçamos o valor da função $f(x)$ em 4 pontos de controle. Podemos interpolar um polinômio de ordem 3 a passar por estes 4 pontos.

i	x	y
0	3.0	5.0
1	4.5	7.1
2	6.0	8.7
3	9.0	10.0

```
--> x=[3.0, 4.5, 6.0, 9.0];

--> A=[(x^0)',(x^1)',(x^2)',(x^3)']
 A  =

   1.   3.    9.       27.
   1.   4.5   20.25    91.125
   1.   6.    36.      216.
   1.   9.    81.      729.

--> y=[5.0; 7.1; 8.7; 10.0];

--> c=inv(A)*y
 c  =

  -0.3000000
   1.9444444
  -0.0444444
  -0.0049383
```

Interpolação cúbico -
polinômio de ordem 3 - $P_3(x)$

$$P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Como o polinômio deve passar nos 4 pontos de controle temos:

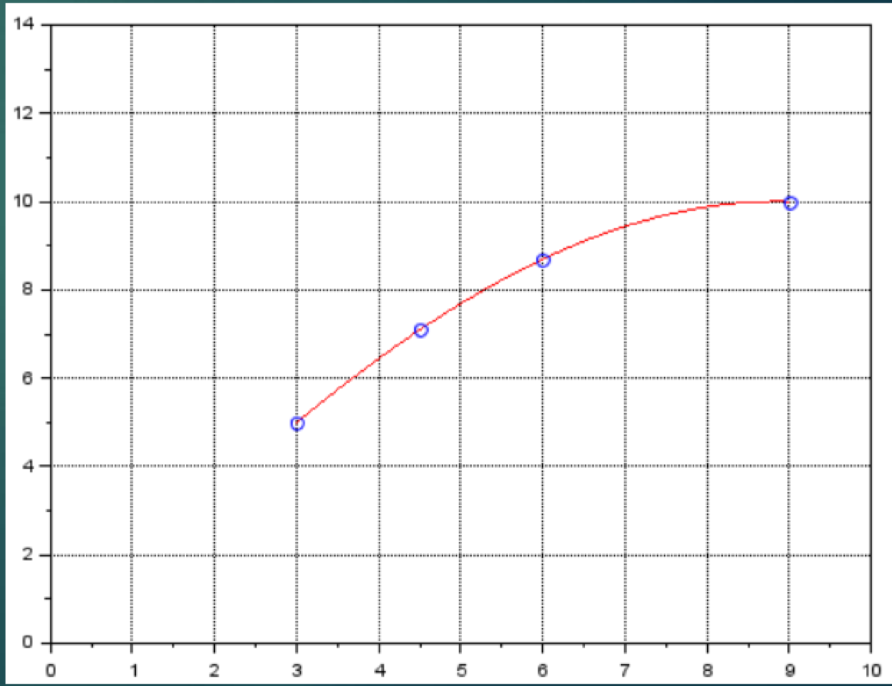
$$P_3(3) = a_0 + a_1(3) + a_2(3)^2 + a_3(3)^3 = 5.0$$
$$P_3(4.5) = a_0 + a_1(4.5) + a_2(4.5)^2 + a_3(4.5)^3 = 7.1$$
$$P_3(6) = a_0 + a_1(6) + a_2(6)^2 + a_3(6)^3 = 8.6$$
$$P_3(9) = a_0 + a_1(9) + a_2(9)^2 + a_3(9)^3 = 10$$

$$\begin{bmatrix} 1 & 3 & 3^2 & 3^3 \\ 1 & 4.5 & 4.5^2 & 4.5^3 \\ 1 & 6 & 6^2 & 6^3 \\ 1 & 9 & 9^2 & 9^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 5.0 \\ 7.1 \\ 8.7 \\ 10.0 \end{bmatrix}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -0.3 \\ 1.9444 \\ -0.0444 \\ -0.0049 \end{bmatrix}$$

$$P_3(x) = -0.3 + 1.9444x - 0.0444x^2 - 0.0049x^3$$

```
--> ps=poly(c,"s","coeff")
 ps  =

  -0.3 +1.9444444s -0.0444444s² -0.0049383s³

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

--> plot(xi,horner(ps,xi),'r')

--> scatter(x,y)

--> xgrid()
```



Polinômio de Vandermonde

Se tivermos (n+1) pontos de controle, haverá um único polinômio $P_n(x)$ de grau n, que passará por todo os pontos $(x_i, f(x_i))$

i	x_i	$f(x_i)$
0	x_o	$f(x_o)$
1	x_1	$f(x_1)$
2	x_2	$f(x_2)$
...	...	...
n	x_n	$f(x_n)$

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \cdots a_nx^n$$

Se tivermos 5 pontos de controle, haverá um único polinômio $P_4(x)$ de grau 4, que passará por todo os pontos $(x_i, f(x_i))$

i	x_i	$f(x_i)$
0	x_o	$f(x_o)$
1	x_1	$f(x_1)$
2	x_2	$f(x_2)$
3	x_3	$f(x_3)$
4	x_4	$f(x_4)$

$$P_4(x_o) = a_0 + a_1x_o + a_2{x_o}^2 + a_3{x_o}^3 + a_4{x_o}^4 = f(x_o)$$
$$P_4(x_1) = a_0 + a_1x_1 + a_2{x_1}^2 + a_3{x_1}^3 + a_4{x_1}^4 = f(x_1)$$
$$P_4(x_2) = a_0 + a_1x_2 + a_2{x_2}^2 + a_3{x_2}^3 + a_4{x_2}^4 = f(x_2)$$
$$P_4(x_3) = a_0 + a_1x_3 + a_2{x_3}^2 + a_3{x_3}^3 + a_4{x_3}^4 = f(x_3)$$
$$P_4(x_4) = a_0 + a_1x_4 + a_2{x_4}^2 + a_3{x_4}^3 + a_4{x_4}^4 = f(x_4)$$

$$[A|b]$$

$$\begin{bmatrix} 1 & x_0 & {x_o}^2 & {x_0}^3 & {x_0}^4 & | & f(x_o) \\ 1 & x_1 & {x_1}^2 & {x_1}^3 & {x_1}^4 & | & f(x_1) \\ 1 & x_2 & {x_2}^2 & {x_2}^3 & {x_2}^4 & | & f(x_2) \\ 1 & x_3 & {x_3}^2 & {x_3}^3 & {x_3}^4 & | & f(x_3) \\ 1 & x_4 & {x_4}^2 & {x_4}^3 & {x_4}^4 & | & f(x_4) \end{bmatrix}$$

$$A^{-1}b = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

$$P_4(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$$

Retonando ao exemplo , agora com 5 pontos

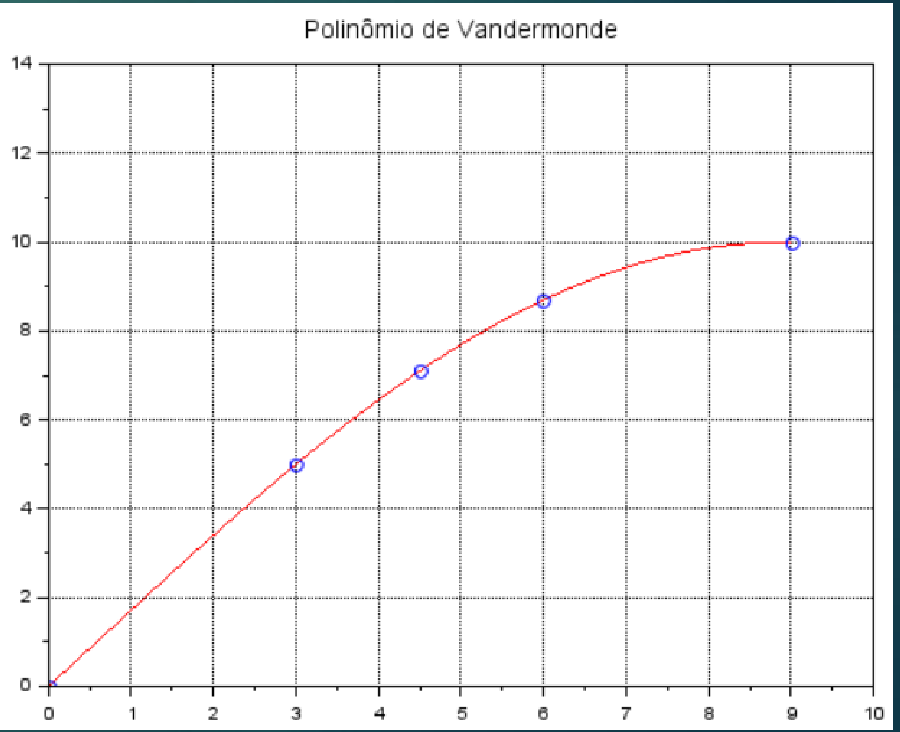
i	x	y
0	0	0
1	3.0	5.0
2	4.5	7.1
3	6.0	8.7
4	9.0	10.0

```
function [pv,A,u]=PolinomioVandermonde(x,y)
    N=length(x);
    A=zeros(N,N)
    for(k=1:N)
        A(:,k)=(x.^(k-1))'
    end
    u=EliminacaoGauss(A,y',%F);
    pv=poly(u,'s','coeff')
endfunction
```

$$P_4(x) = 1.694x\ + 0.02963x^2\ - 0.0142x^3\ + 0.0004115s^4$$

```
function Plot_PolinomioVandermonde(x,y)
    [pv,A,coef]=PolinomioVandermonde(x,y);
    printf("Matriz Aumentada [A|y]")
    disp([A,y'])
    printf("Coeficientes u=A^-1 * y")
    disp(coef)
    printf("Polinômio de Vandermonde\npv(s)=")
    disp(pv)
    xi=linspace(min(x),max(x),1000);
    plot(xi,horner(pv,xi),'r')
    scatter(x,y)
    xgrid()
    xtitle("Polinômio de Vandermonde");
endfunction
```

```
--> x=[0,3,4.5,6,9];

--> y=[0.0,5.0,7.1,8.7,10.0];

--> Plot_PolinomioVandermonde(x,y);
Matriz Aumentada [A|y]
   1.   0.    0.       0.        0.          0.
   1.   3.    9.       27.       81.         5.
   1.   4.5   20.25    91.125    410.0625    7.1
   1.   6.    36.      216.      1296.       8.7
   1.   9.    81.      729.      6561.       10.
Coeficientes u=A^-1 * y
   0.
   1.6944444
   0.0296296
  -0.0141975
   0.0004115
Polinômio de Vandermonde
pv(s)=
  1.6944444s +0.0296296s² -0.0141975s³ +0.0004115s⁴
```



Problemas com Vandermonde

- Notar que, com o aumento do número de pontos, será necessária a resolução de um sistema de ordem mais alta.

- A resolução de sistemas de equações sempre envolve a inversão da matriz característica A, o que gera erros de arredondamento. O uso de Eliminação de Gauss pode ajudar.

- Além da ordem mais alta da matriz característica, haverá uma disparidade grande na ordem de grandeza dos elementos da matriz A, na primeira coluna temos valor 1, na última coluna x^n, o que pode gerar erros de arredondamento consideráveis.

- Notar também que os coeficientes do polinômio, principalmente nas potências mais elevada, serão muito pequenos, o que pode gerar perda de mantissa em somas ($2.6\ 7\ 10^{-9}$).

Polinômio de Newton

i	x_i	$f(x_i)$
0	x_o	$f(x_o)$
1	x_1	$f(x_1)$
2	x_2	$f(x_2)$
…	…	…
n	x_n	$f(x_n)$

- Os erros de arredondamento em Vandermonde são decorrentes principalmente do cálculo dos coeficientes do polinômio, que é feito através da inversão de uma matriz de ordem n (para n+1 pontos de controle)
- Uma maneira de contornar este problema é utilizar o esquema de cálculo dos polinômios de Newton.
- Neste esquema, os coeficientes do polinômio são encontrados diretamente de uma tabela de diferenças finitas.
- Notar que, do ponto de vista puramente algébrico, o polinômio de Vandermonde e o polinômio de Newton são exatamente os mesmos.
- No entanto, como temos resolução finita em um computador, os erros de Vandermonde são maiores que os erros de Newton.

Se tivermos (n+1) pontos de controle, haverá um único polinômio $P_n(x)$ de grau n, que passará por todo os pontos $(x_i, f(x_i))$

$$\mathrm{P_n(x) = b_o + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2) + \cdots + b_n(x - x_0)(x - x_1) \ldots (x - x_{n-1})}$$

onde

$$\begin{aligned}
b_o &= f(x_0)\\
b_1 &= f[x_1, x_0]\\
b_2 &= f[x_2, x_1, x_0]\\
b_3 &= f[x_3, x_2, x_1, x_0]\\
&\ldots\\
b_n &= f[x_n, x_{n-1}, \ldots, x_1, x_0]
\end{aligned}$$

i	x	y	Δ1	Δ2	Δ3	Δ4
0	0	0				
1	3	5	1.66667			
2	4.5	7.1	1.40000	-0.05926		
3	6	8.7	1.06667	-0.11111	-0.00864	
4	9	10	0.43333	-0.14074	-0.00494	0.00041

$$b_0 = f(x_o) = 0.0$$

$$b_1 = \Delta_1 = \frac{(5-0)}{3-0} = 1.6666667$$

$$b_2 = \Delta_2 = \frac{(1.400000\ - 1.6667)}{4.5-0} = -0.0592593$$

$$b_3 = \Delta_3 = \frac{(-0.1111111 - (-0.0592593))}{6-0} = -0.008642$$

$$b_4 = \Delta_4 = \frac{-0.0049383 - (-0.008642))}{9-0} = 0.0004115$$

$$P_4(x) = 0 + 1.66667(x-0) - 0.05926(x-0)(x-3) - 0.00864(x-0)(x-3)(x-4.5) + 0.00041(x-0)(x-3)(x-4.5)(x-6)$$

i	x	y
0	0	0
1	3.0	5.0
2	4.5	7.1
3	6.0	8.7
4	9.0	10.0

```
function [b,D]=PolinomioNewton(x,y)
    N=length(x);    //Cálculo dos coeficientes
    D(:,1)=y';
    for j=2:N   // D(N,N) - Tabela de diferenças
        for k=j:N
            D(k,j)=(D(k,j-1)-D(k-1,j-1))/(x(k)-x(k-j+1));
        end
    end
    b=diag(D)
endfunction
```

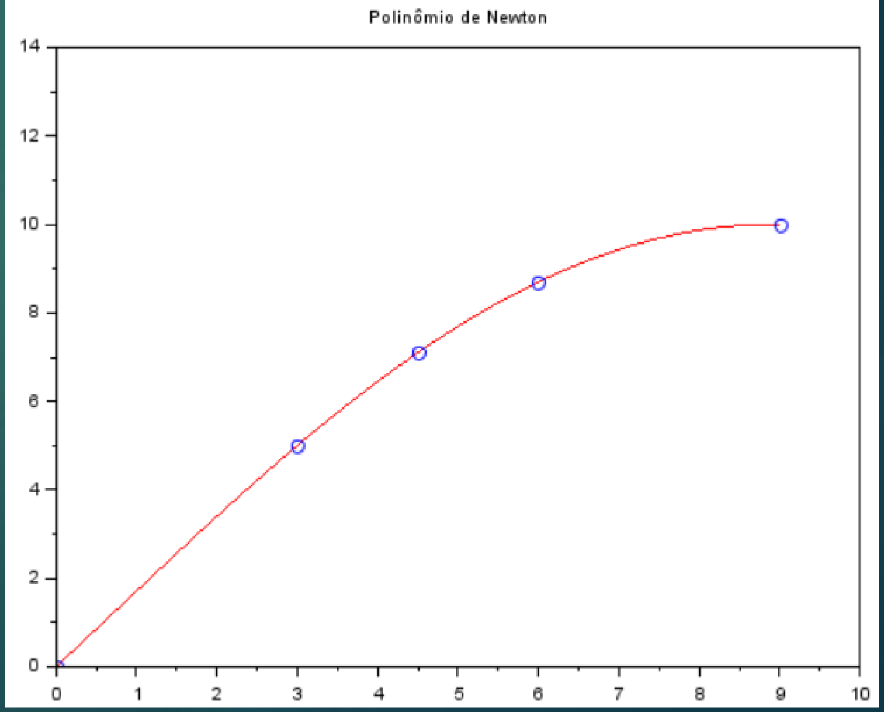
```
--> x=[0,3,4.5,6,9];

--> y=[0.0,5.0,7.1,8.7,10.0];

--> b=PolinomioNewton(x,y)
 b  =

   0.
   1.6666667
  -0.0592593
  -0.0086420
   0.0004115
```

```
function yp=InterNewton(xp,x,b)
  N=length(b)
  yp=b(N);
  for k=(N-1):-1:1
      yp=yp.*(xp-x(k)) + b(k);
  end
endfunction
```

```
function Plot_PolinomioNewton(x,y)
  [coef,D]=PolinomioNewton(x,y)
  N=length(x)
  printf("Tabela de diferenças")
  disp(D)
  printf("Coeficientes b=diag(D)")
  disp(coef)
  printf("Polinômio de Newton\npn(s)=\n")
  for k=(N-1):-1:1
    printf("(%.5e)",coef(N-k+1))
    for(j=N-1:-1:k) printf("(x-(%.5f))",x(N-j)) end
    if (k>1) then printf(" + \n") end
  end
  xi=linspace(min(x),max(x),1000);
  plot(xi,InterNewton(xi,x,coef),'r')
  scatter(x,y)
  xgrid()
  xtitle("Polinômio de Newton")
endfunction
```

```
--> x=[0,3,4.5,6,9];

--> y=[0.0,5.0,7.1,8.7,10.0];

--> Plot_PolinomioNewton(x,y);
Tabela de diferenças
   0.     0.          0.          0.          0.
   5.     1.6666667   0.          0.          0.
   7.1    1.4        -0.0592593   0.          0.
   8.7    1.0666667  -0.1111111  -0.008642    0.
   10.    0.4333333  -0.1407407  -0.0049383   0.0004115
Coeficientes b=diag(D)
   0.
   1.6666667
  -0.0592593
  -0.0086420
   0.0004115
Polinômio de Newton
pn(s)=
(1.66667e+00)(x-(0.00000)) +
(-5.92593e-02)(x-(0.00000))(x-(3.00000)) +
(-8.64198e-03)(x-(0.00000))(x-(3.00000))(x-(4.50000)) +
(4.11523e-04)(x-(0.00000))(x-(3.00000))(x-(4.50000))(x-(6.00000))
```

$$P_4(x) = 0 + 1.66667(x) - 0.05926(x)(x-3) - 0.00864(x)(x-3)(x-4.5) + 0.00041(x)(x-3)(x-4.5)(x-6)$$



- Para o cálculo dos coeficientes do polinômio de Newton, não precisamos inverter uma matriz (como em Vandermonde), precisamos simplesmente completar uma tabela de diferenças.
- Notar que o polinômio simplificado de Newton é o mesmo que o polinômio de Vandermonde, somente escito de forma diversa, havendo diferenças apenas no arredondamento do cálculo dos coeficientes.

- Para avaliar o polinômio de Newton em um ponto específico, devemos utilizar a fórmula completa com os coeficientes b's, e não o polinômio simplificado, pois se utilizarmos o polinômio simplificado perderemos a vantagem numérica do cálculo de Newton.

```
Polinômio de Newton
pn(s)=
(1.66667e+00)(x-(0.00000)) +
(-5.92593e-02)(x-(0.00000))(x-(3.00000)) +
(-8.64198e-03)(x-(0.00000))(x-(3.00000))(x-(4.50000)) +
(4.11523e-04)(x-(0.00000))(x-(3.00000))(x-(4.50000))(x-(6.00000))
```

- Há ainda problemas de arredondamento decorrentes dos valores muito pequenos dos coeficientes.
- No entanto, o erros de arredondamento de Newton são menores que os de Vandermonde.

Polinômio de Lagrange

- Com o polinômio de Lagrange, eliminamos a necessidade de se calcular coeficientes.
- Para tanto utilizamos a Base de Lagrange:

i	x_i	$f(x_i)$
0	x_o	$f(x_o)$
1	x_1	$f(x_1)$
2	x_2	$f(x_2)$
...	...	...
n	x_n	$f(x_n)$

$$L_i^n(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}$$

$$P_n(x) = \sum_{i=1}^{n} L_i^n(x) f(x_i)$$

Observar que :

$$L_i^n(x_k) = \begin{cases} 1 \; se \; k = i \\ 0 \; se \; k \neq i \end{cases}$$

i	x	y
0	0	0
1	3.0	5.0
2	4.5	7.1
3	6.0	8.7
4	9.0	10.0

$$L_0^4 = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)} = \frac{(x-3)(x-4.5)(x-6)(x-9)}{(0-3)(0-4.5)(0-6)(0-9)}$$

$$L_1^4 = \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)} = \frac{(x-0)(x-4.5)(x-6)(x-9)}{(3-0)(3-4.5)(3-6)(3-9)}$$

$$L_2^4 = \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)} = \frac{(x-0)(x-3)(x-6)(x-9)}{(4.5-0)(4.5-3)(4.5-6)(4.5-9)}$$

$$L_3^4 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)} = \frac{(x-0)(x-3)(x-4.5)(x-9)}{(6-0)(6-3)(6-4.5)(6-9)}$$

$$L_4^4 = \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)} = \frac{(x-0)(x-3)(x-4.5)(x-6)}{(9-0)(9-3)(9-4.5)(0-6)}$$

$$P_4(x) = y_o\, L_0^4(x) + y_1\, L_1^4(x) + y_2\, L_2^4(x) + y_3\, L_3^4(x) + y_4\, L_4^4(x)$$

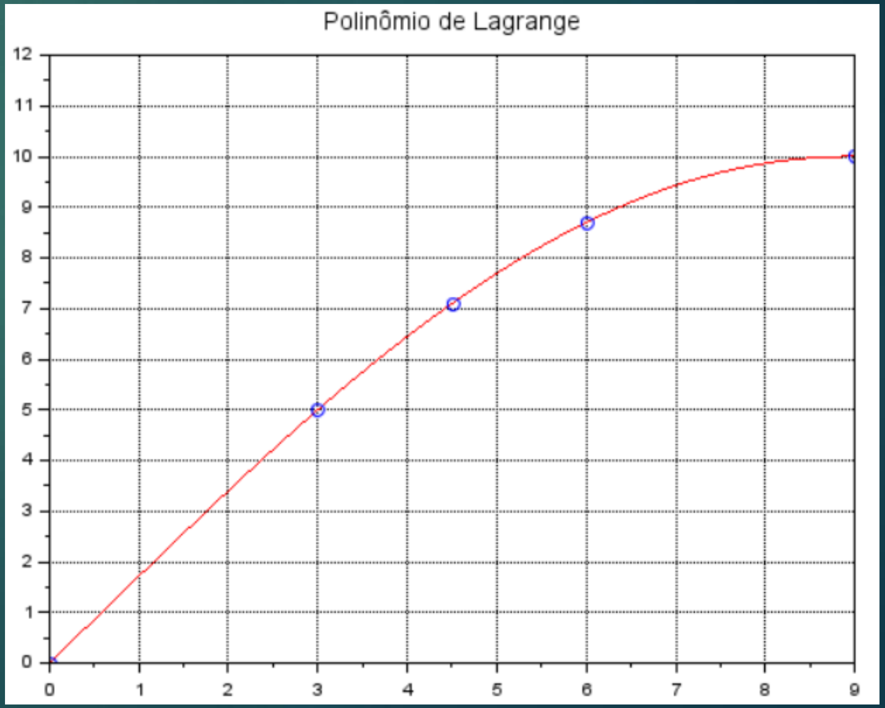
i	x	y
0	0	0
1	3.0	5.0
2	4.5	7.1
3	6.0	8.7
4	9.0	10.0

```
function yp=InterLagrange(xp,x,y)
    N=length(x)
    M=length(xp)
    for k=1:M
        Li = y
        for (i=1:N)
            for j=[1:i-1 i+1:N]
                Li(i)=Li(i)*(xp(k)-x(j))/(x(i)-x(j));
            end
        end
        yp(k)=sum(Li)
    end
endfunction
```

```
--> x=[0.0, 3.0, 4.5, 6.0, 9.0];

--> y=[0.0, 5.0, 7.1, 8.7, 10.0];

--> Plot_PolinomioLagrange(x,y)
Polinômio de Lagrange
L[0,4](x)=
        (x-(3.0000))(x-(4.5000))(x-(6.0000))(x-(9.0000))
(0.0000)---------------------------------------------------
        (-3.0000)(-4.5000)(-6.0000)(-9.0000)
L[1,4](x)=
        (x-(0.0000))(x-(4.5000))(x-(6.0000))(x-(9.0000))
(5.0000)---------------------------------------------------
        (3.0000)(-1.5000)(-3.0000)(-6.0000)
L[2,4](x)=
        (x-(0.0000))(x-(3.0000))(x-(6.0000))(x-(9.0000))
(7.1000)---------------------------------------------------
        (4.5000)(1.5000)(-1.5000)(-4.5000)
L[3,4](x)=
        (x-(0.0000))(x-(3.0000))(x-(4.5000))(x-(9.0000))
(8.7000)---------------------------------------------------
        (6.0000)(3.0000)(1.5000)(-3.0000)
L[4,4](x)=
        (x-(0.0000))(x-(3.0000))(x-(4.5000))(x-(6.0000))
(10.0000)---------------------------------------------------
        (9.0000)(6.0000)(4.5000)(3.0000)
pl=y(0)*L[0,4](x)+y(1)*L[1,4](x)+y(2)*L[2,4](x)+y(3)*L[3,4](x)+y(4)*L[4,4](x)
```

$$P_4(x) = 0\ \ L_0^4(x) + 5.0\ \ L_1^4(x) + 7.1\ \ L_2^4(x) + 8.7\ L_3^4(x) + 10.0\ L_4^4(x)$$

```
function  Plot_PolinomioLagrange(x,y)
  N=length(x);
  printf("Polinômio de Lagrange\n")
  s=poly(0,'s');
  linha_s = "---------------------------------------------------"
  for i=1:N
      num_s = ""
      den_s = ""
      for j=[1:N]
          if (j<>i)
              num_s=sprintf("%s(x-(%.4f))",num_s,x(j))
              den_s=sprintf("%s(%.4f)",den_s,x(i)-x(j))
          end
      end
      printf("L[%d,%d](x)=\n\t%s\n(%.4f)%s\n\t%s\n",i-1,N-1,num_s,y(i),linha_s,den_s)
  end
  printf("pl=")
  for i=1:N-1  printf("y(%d)*L[%d,%d](x)+",i-1,i-1,N-1) end
  printf("y(%d)*L[%d,%d](x)\n",N-1,N-1,N-1)
  xi=linspace(min(x),max(x),1000);
  plot(xi,InterLagrange(xi,x,y)','r')
  scatter(x,y)
  xgrid()
  xtitle("Polinômio de Lagrange");
endfunction
```



- No método de Lagrange, não é necessário o cálculo de coeficientes, o polinômio é montado diretamente dos pontos amostrais.
- Notar que o polinômio de Lagrange simplificado é praticamente o mesmo que os polinômios de Vandermonde e Newton, havendo diferenças apenas no arredondamento do cálculo dos coeficientes.

- Para interpolar o polinômio de Lagrange em um ponto intermediário, devemos utilizar a fórmula completa do polinômio, e não o polinômio simplificado, pois se utilizarmos o polinômio simplificado perderemos a vantagem numérica do cálculo de Lagrange.

```
L[0,4](x)=
          (x-(3.0000))(x-(4.5000))(x-(6.0000))(x-(9.0000))
(0.0000)----------------------------------------------------
          (-3.0000)(-4.5000)(-6.0000)(-9.0000)
L[1,4](x)=
          (x-(0.0000))(x-(4.5000))(x-(6.0000))(x-(9.0000))
(5.0000)----------------------------------------------------
          (3.0000)(-1.5000)(-3.0000)(-6.0000)
L[2,4](x)=
          (x-(0.0000))(x-(3.0000))(x-(6.0000))(x-(9.0000))
(7.1000)----------------------------------------------------
          (4.5000)(1.5000)(-1.5000)(-4.5000)
L[3,4](x)=
          (x-(0.0000))(x-(3.0000))(x-(4.5000))(x-(9.0000))
(8.7000)----------------------------------------------------
          (6.0000)(3.0000)(1.5000)(-3.0000)
L[4,4](x)=
          (x-(0.0000))(x-(3.0000))(x-(4.5000))(x-(6.0000))
(10.0000)----------------------------------------------------
          (9.0000)(6.0000)(4.5000)(3.0000)
pl=y(0)*L[0,4](x)+y(1)*L[1,4](x)+y(2)*L[2,4](x)+y(3)*L[3,4](x)+y(4)*L[4,4](x)
```

```
--> x=[0,3,4.5,6,9];

--> y=[0,5.0,7.1,8.7,10.0];

--> pv=PolinomioVandermonde(x,y);

--> horner(pv,5)
 ans  =

   7.6954732510288064162296

--> coef=PolinomioNewton(x,y);

--> InterNewton(5,x,coef)
 ans  =

   7.6954732510288064162296

--> InterLagrange(5,x,y)
 ans  =

   7.6954732510288064162296
```

- Os erros de arredondamento de Lagrange são um pouco melhores que os de Vandermonde e de Newton