UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 68993
ANDRE HEINECK HAHN 19FCN117 - 1 apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

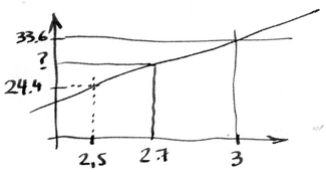
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   69  356.2
   94 1855.5
   ```

   e calculando a interpolação para $x = 87$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    50  256.2
    83 1077
   116 4192.4
   ```

   e calculando a interpolação para $x = 93$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    68  243.9
    97  668.2
   126 1689.4
   155 4119.9
   184 9865.9
   ```

   e calculando a interpolação para $x = 131$

4. Finalmente, um caso para gente grande:

   ```
    54   195
    75   451.7
    96   926.3
   117  1794.4
   138  3368.1
   159  6201
   180 11275.8
   201 20334.5
   ```

   e calculando a interpolação para $x = 130$

1	2	3	4

117-68993 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69789
BRUNO V GUIMARARES 19FCN117 - 2 apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

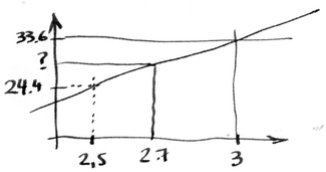
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   67  485.9
   99 3648.6
   ```

   e calculando a interpolação para $x = 181$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    67  321.7
   107 1115.3
   147 3834.9
   ```

   e calculando a interpolação para $x = 108$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    59   267.6
    98   736.2
   137  1872.2
   176  4595.5
   215 11079.9
   ```

   e calculando a interpolação para $x = 284$

4. Finalmente, um caso para gente grande:

   ```
    51   159
    75   331.6
    99   637.6
   123  1176.1
   147  2117.9
   171  3757
   195  6599.6
   219 11517.4
   ```

   e calculando a interpolação para $x = 324$

1	2	3	4

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69008
EDUARDO SCARANTE DEMCZUK 19FCN117 - 3 apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

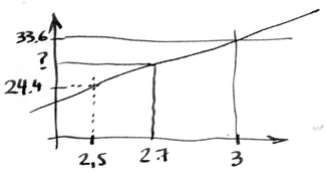
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   50  485.9
   74 3307.7
   ```

   e calculando a interpolação para $x = 120$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    44  279.5
    76 1169.6
   108 4560.4
   ```

   e calculando a interpolação para $x = 83$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    55   251.2
    78   750.7
   101  1998.9
   124  5069.5
   147 12548.5
   ```

   e calculando a interpolação para $x = 212$

4. Finalmente, um caso para gente grande:

   ```
    43   177.7
    66   440.6
    89   933.3
   112  1844.9
   135  3514.1
   158  6545.8
   181 12021
   204 21868.1
   ```

   e calculando a interpolação para $x = 257$

1	2	3	4

117-69008 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69015
FELIPE DE ROSSI REZENDE 19FCN117 - 4
apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

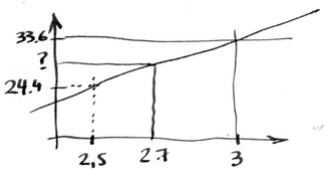
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
    68  422.1
   104 2639.7
   ```

   e calculando a interpolação para $x = 194$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    45  378.2
    81 1524.7
   117 5886.1
   ```

   e calculando a interpolação para $x = 84$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    45  199
    69  549.5
    93 1385.6
   117 3356.8
   141 7970
   ```

   e calculando a interpolação para $x = 232$

4. Finalmente, um caso para gente grande:

   ```
    51   137.6
    72   308.1
    93   614.6
   114  1160.7
   135  2126.1
   156  3822
   177  6788.5
   198 11960.6
   ```

   e calculando a interpolação para $x = 264$

1	2	3	4

117-69015 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69022
GIOVANA STOPANOVSKI BECKER 19FCN117 - 5 apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

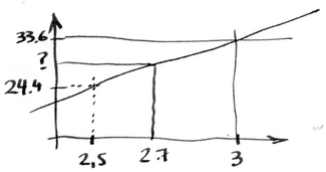
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   53  477.3
   77 2999.8
   ```

   e calculando a interpolação para $x = 127$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    57  284.7
    86  934.4
   115 3065
   ```

   e calculando a interpolação para $x = 158$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    46  227.7
    75  601.4
   104 1483
   133 3543.3
   162 8332.3
   ```

   e calculando a interpolação para $x = 110$

4. Finalmente, um caso para gente grande:

   ```
    55   218.3
    76   494
    97  1003.6
   118  1935.4
   139  3624.5
   160  6666.9
   181 12121.4
   202 21868.1
   ```

   e calculando a interpolação para $x = 312$

1	2	3	4

117-69022 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69208
IAN DO AMARAL PIMENTA 19FCN117 - 6
apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

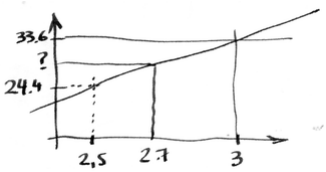
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   59  533.7
   96 3375
   ```

   e calculando a interpolação para $x = 184$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
   44  337.7
   68 1458.4
   92 5859
   ```

   e calculando a interpolação para $x = 70$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    36   298.9
    58   794.3
    80  1983.9
   102  4815
   124 11517.4
   ```

   e calculando a interpolação para $x = 190$

4. Finalmente, um caso para gente grande:

   ```
    64   177.7
    94   394.4
   124   788.3
   154  1497.7
   184  2764.8
   214  5014.6
   244  8992
   274 16001.5
   ```

   e calculando a interpolação para $x = 177$

1	2	3	4

117-69208 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69039
KHAREN INGRID VIDAL BELO 19FCN117 - 7
apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

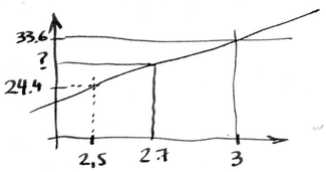
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   40  320.2
   62 1818.8
   ```

   e calculando a interpolação para $x = 109$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    47  210.4
    78  832.6
   109 3086.8
   ```

   e calculando a interpolação para $x = 84$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    55   204.4
    78   636.4
   101  1718.4
   124  4378.9
   147 10846.6
   ```

   e calculando a interpolação para $x = 107$

4. Finalmente, um caso para gente grande:

   ```
    58   189.2
    98   361.8
   138   660.7
   178  1176.1
   218  2061.2
   258  3576.5
   298  6165.3
   338 10582.1
   ```

   e calculando a interpolação para $x = 205$

1	2	3	4

117-69039 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69046
MARCELL ANDREY MACHADO PINTO
19FCN117 - 8 apos 28/05, 50% _____ / _____ /

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

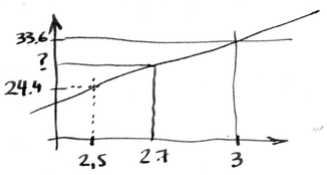
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   39  636.5
   68 4535.5
   ```

   e calculando a interpolação para $x = 138$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    53  303.1
    77 1200.9
   101 4535.5
   ```

   e calculando a interpolação para $x = 84$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    64   295.2
   102   834.6
   140  2165.7
   178  5409.1
   216 13252
   ```

   e calculando a interpolação para $x = 282$

4. Finalmente, um caso para gente grande:

   ```
    26  167.6
    50  318.8
    74  578.8
    98 1024
   122 1783.3
   146 3074.5
   170 5265.6
   194 8979
   ```

   e calculando a interpolação para $x = 284$

1	2	3	4

117-69046 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69060
MELISSA SCHELLIN BECKER 19FCN117 - 9
apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

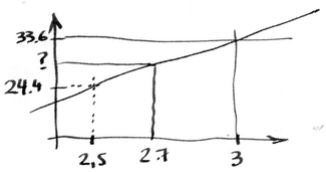
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
    65  494.5
   100 3330.1
   ```

   e calculando a interpolação para $x = 149$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
   27  170.9
   49  631.7
   71 2216.5
   ```

   e calculando a interpolação para $x = 125$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    58  177.8
    98  468.1
   138 1142.5
   178 2693.7
   218 6241.7
   ```

   e calculando a interpolação para $x = 145$

4. Finalmente, um caso para gente grande:

   ```
    57   147.6
    94   303.9
   131   578.8
   168  1059.1
   205  1893.3
   242  3335.4
   279  5820.3
   316 10092
   ```

   e calculando a interpolação para $x = 192$

1	2	3	4

117-69060 - 28/05

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69077
TAYANE CORDEIRO FARIA 19FCN117 - 10
apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

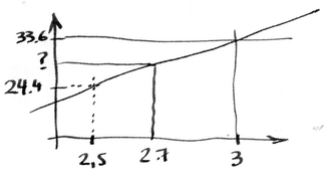
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   36  409.6
   60 2177.5
   ```

   e calculando a interpolação para $x = 52$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    56  230.6
    87  775.8
   118 2577.7
   ```

   e calculando a interpolação para $x = 90$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    50   278.6
    83   739.8
   116  1842.7
   149  4455.9
   182 10615
   ```

   e calculando a interpolação para $x = 231$

4. Finalmente, um caso para gente grande:

   ```
    63   200.8
    98   418.5
   133   808.8
   168  1503.1
   203  2730.2
   238  4888.7
   273  8672.3
   308 15288.7
   ```

   e calculando a interpolação para $x = 412$

1	2	3	4

UFPR - UP - UTFPR - PUCPr – 04/05/2019 - 21:20:03.5
Prof Dr P Kantek (pkantek@gmail.com)
Interpolação Polinomial VIVO739p, V: 1.05 69796
THONNY G FERRAZ 19FCN117 - 11 apos 28/05, 50% _____ / _____ / _____

Interpolação polinomial

Suponha que você está estudando um fenômeno físico qualquer e ocorre uma destas duas situações:

- Não se tem idéia da expressão analítica (fórmula) que descreve o fenômeno, e o que se tem são apenas pares de valores levantados *in loco* (no campo).
- A fórmula é conhecida, mas é extraordinária complicada, eventualmente descontínua o que inviabiliza derivadas e integrais a ela associadas.

Uma proposta de tratamento para esta classe de problemas é procurar um polinômio que se aproxime tanto quanto possível (e desejado) da função desconhecida original. Agora, todo o tratamento analítico se dá sobre o polinômio aproximado e não sobre a função original (que lembrando, ou é desconhecida ou é muito complexa). No primeiro caso viabiliza-se o tratamento e no segundo troca-se precisão por simplicidade no manuseio do problema.

Algumas vantagens deste polinômio:

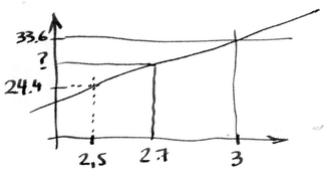
- polinômios são facilmente computáveis;
- suas derivadas e suas integrais continuam sendo polinômios;
- é fácil (e já é conhecido) achar suas raízes;

Um exemplo

Suponha um sistema hidráulico para transporte de água. Suas perdas supostamente são proporcionais à vazão do sistema. Em campo, levantaram-se os seguintes dados

vazão em m^3/h	perdas em %
1.5	10
2	16.5
2.5	24.2
3	33.6
3.5	44
4	55.6

A questão que se poderia levantar aqui é **Qual seria a perda do sistema em uma vazão de** $2.7m^3/h$ **?** Desenhando isto num gráfico poder-se-ia ter



Por semelhança de triângulos pode-se escrever

$$\frac{\Delta y}{\Delta x} = \frac{33.6 - 24.4}{3 - 2.5} = \frac{y - 24.4}{2.7 - 2.5}$$

e daqui $y = 28.08$.

Como consideraram-se 2 pontos a aproximação obtida é linear. A regra da interpolação é

para n pontos $\Rightarrow$ polinômio grau $n-1$

Neste caso o polinômio linear (uma reta) é

$$P_1 = a_0 + a_1 x$$

onde a_0 é o coeficiente linear da reta e a_1 é o angular. Aplicando-se esta abordagem a dois pontos, tem-se
$a_0 + a_1 x_1 = y_1$
$a_0 + a_1 x_2 = y_2$
Ou na forma matricial

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Que pode ser resolvido por qualquer dos métodos já conhecidos. Aqui vai-se usar o método de escalonamento de Gauss. Por exemplo:

$$\begin{bmatrix} 1 & 2.5 \\ 1 & 3 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} 24.4 \\ 33.6 \end{Bmatrix}$$

o que dará $a_0 = -21.6$ e $a_1 = 18.4$. Daqui, $P_1(2.7) = 28.08$.

Ao refazer o mesmo exemplo para 3 pontos, vai-se criar agora um polinômio de grau 2 (3-1=2) no formato

$$P_2(x) = a_2 x^2 + a_1 x + a_0$$

e o sistema fica

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \cdot \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}$$

Python

```
def intpol(a,y):
    import numpy as np
    n=len(a)
    x=np.zeros((n),float)
    passo=0
    while passo<n-1:
        i=passo+1
        while i<n:
            pivot=a[i,passo]/a[passo,passo]
            j=0
            while j<n:
                a[i,j]=a[i,j]-pivot*a[passo,j]
                j=j+1
            y[i]=y[i]-pivot*y[passo]
            i=i+1
        passo=passo+1
    x[n-1]=y[n-1]/a[n-1,n-1]
    i=n-1
    while i>=0:
        x[i]=y[i]
        j=i+1
        while j<n:
            x[i]=x[i]-a[i,j]*x[j]
            j=j+1
        x[i]=x[i]/a[i,i]
        i=i-1
    xint = float(input('Qual o valor a interpolar ? '))
    yint=0.0
    i=0
    while i<n:
        yint=yint+(x[i]*xint**(i))
        i=i+1
    print('y interpolado ',yint)

import numpy as np
a=np.array([[1,49,0,0,0],[1,75,0,0,0],[1,101,0,0,0],
            [1,127,0,0,0],[1,153,0,0,0]],float)
b=np.array([289.6, 776, 1946.5, 4736.3, 11348.5],float)
i=0
while i<5:                    #importante: este trecho
    j=2                       # e' opcional e serve
    while j<5:                # para preencher x^2,
        a[i,j]=a[i,1]**j      # x^3 e x^4 que ficaram
        j=j+1                 # zerados lá na entrada
    i=i+1                     # de dados... veja la'
intpol(a,b)
```

☞ Para você fazer

Observação importante: em caso de números não inteiros, trabalhe com apenas 1 casa decimal.

1. Neste primeiro exemplo, você deve resolver **à mão** devolvendo os cálculos junto a interpolação do polinômio

   ```
   43  352.2
   72 2138.8
   ```

   e calculando a interpolação para $x = 142$

2. Novamente, resolvendo **a mão** (e entregando os cálculos junto) ache a interpolação do polinômio

   ```
    82  383.7
   120 1685
   158 6887.5
   ```

   e calculando a interpolação para $x = 219$

3. Agora, pode e deve usar o programa Python e calcular a interpolação para

   ```
    68  242.1
    99  625.9
   130 1525.2
   161 3615.9
   192 8454.2
   ```

   e calculando a interpolação para $x = 260$

4. Finalmente, um caso para gente grande:

   ```
    58   186.3
    88   451.7
   118   947.3
   148  1861.8
   178  3532.4
   208  6560.9
   238 12021
   268 21826.1
   ```

   e calculando a interpolação para $x = 311$

1	2	3	4