**TRABALHO COMPUTACIONAL 4**

Nomes:

* Daiana Santos 120.357
* Isadora Muniz 120.431
* Luciana Bello 120.506
* Maria Victória Siqueira 120.529

# 

# **Parte 1**

**Exercício 1**

Como queremos um polinômio de grau menor ou igual a 5, definimos o polinômio como:

Tabela de diferenças divididas:

| w | Ordem 0 | Ordem 1 | Ordem 2 | Ordem 3 | Ordem 4 | Ordem 5 |
| --- | --- | --- | --- | --- | --- | --- |
| 5 | 1226 |  |  |  |  |  |
| 10 | 1498 | 54,4 |  |  |  |  |
| 15 | 1822 | 64,8 | 1,04 |  |  |  |
| 20 | 2138 | 63,2 | -0,16 | -0,08 |  |  |
| 30 | 2662 | 52,4 | -0,72 | -0,028 | 0,00208 |  |
| 40 | 2840 | 17,8 | -1,73 | -0,0404 | -0,0004134 | -0,000071238 |

Substituindo na expressão:

Simplificando a equação:

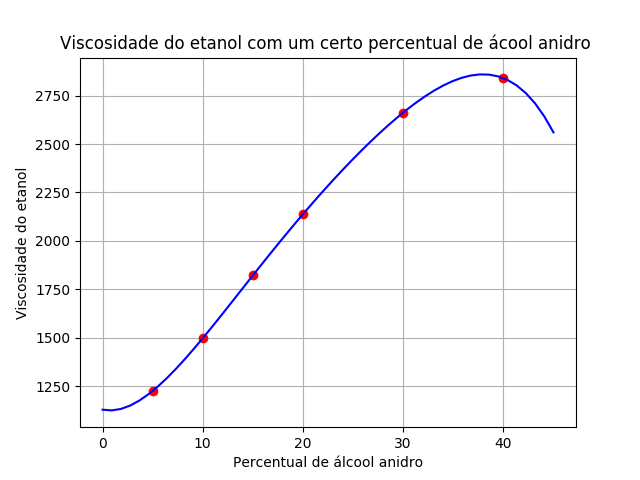
Para os pontos 12, 25 e 38:

**P(12) = 1625, 849**

**P(25) = 2422,5428**

**P(38) = 2859,66**

O gráfico que representa o polinômio encontrado está abaixo, com os pontos dados no problema, gerado através do código que está em anexo:



**Vantagens:**

* As contas são mais simples

**Desvantagens:**

* Os resultados da tabela de diferenças estão interligados, o que torna mais complexo por ter que tomar cuidado em não errar nenhum dado da tabela, pois errando um dado, erram todos em sequência.
* Há mais contas.
* Por ter muitas divisões resultou em muitos números decimais.

**Exercício 2**

Como queremos um polinômio de grau menor ou igual a 5, definimos:

Tabela de diferenças divididas:

| w | Ordem 0 | Ordem 1 | Ordem 2 | Ordem 3 | Ordem 4 | Ordem 5 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 14,5 |  |  |  |  |  |
| 2 | 19,5 | 5 |  |  |  |  |
| 3 | 30,5 | 11 | 3 |  |  |  |
| 4 | 53,5 | 23 | 6 | 1 |  |  |
| 5 | 94,5 | 41 | 9 | 1 | 0 |  |
| 6 | 159,5 | 65 | 12 | 1 | 0 | 0 |

Substituindo os valores na expressão:

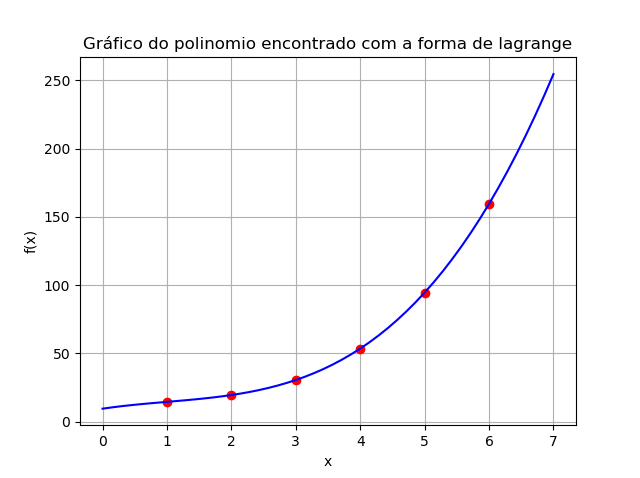
Simplificando a expressão:

Para o ponto 4,5:

**P(4,5) = 71,375**

Concluindo que pode-se obter o mesmo resultado que o problema sugeriu.

O gráfico que representa o polinômio encontrado está abaixo, com os pontos dados no problema, gerado através do código que está em anexo:



**Vantagens:**

* Neste exercício as contas saíram mais simples, conseguindo encontrar o polinômio com mais facilidade.

**Desvantagens:** Não teve desvantagens, já que em todos os aspectos o método de Newton foi mais fácil e rápido.

**Exercício 3**

Como queremos um polinômio de primeira ordem, temos:

**(1)**

Montando a tabela de diferenças divididas, temos:

| x | Ordem 0 | Ordem 1 |
| --- | --- | --- |
| 2 | 7,2 |  |
| 4,25 | 7,1 | -0,0444... |

Substituindo os valores na expressão (1), temos:

Para encontrar um valor de y quando x=4, basta substituir na equação acima. Portanto, temos que: **x=4, y=7,1111111...**

**Vantagens:**

* Em comparação com o método de Lagrange, a forma de Newton foi mais vantajosa pois para encontrar o polinômio de primeira ordem foi mais fácil.

**Desvantagens:**

* O método não apresentou desvantagens comparado ao método de Lagrange.

# **Parte 2**

### **Exercício 1**

1. Para o ajuste da curva, considerou-se as seguintes funções g1(x), g2(x), valores esses baseados em uma curva dada por y = a + b\*x:

* g1(x) = 1
* g2(x) = x

Os valores das constantes a1 e b1 de modo que a função φ(x) = a1\*g1(x) + b1\*g2(x) se aproxime ao máximo de f(x) são dados por:

a1 + =

a1 + =

Assim, tem-se o seguinte sistema:

8\*a1 + 36\*b1 = 9,2

36\*a1 + 204\*b1 = 50,5

Resolvendo o sistema, os valores de a1 e b1 são : a1 = 0,175 ; b1 = 0,216

Sendo assim, **φ(x) = 0,175 + 0,216x**

1. Para o ajuste da curva, considerou-se as seguintes funções g1(x), g2(x), g3(x), valores esses baseados em uma curva dada por y = a + bx + cx²:

* g1(x) = 1
* g2(x) = x
* g3(x) = x²

Os valores das constantes a2, b2 e c2, de modo que a função φ(x) = a2\*g1(x) + b2\*g2(x) + c2\*g3(x) se aproxime ao máximo de f(x), dado por:

a2 ++=

a2 ++=

a2 ++=

Assim, tem-se o seguinte sistema:

8\*a2 + 36\*b2 + 204\*c2 = 9,2

36\*a2 + 204\*b2 + 1296\*c2 = 50,5

204\*a2 + 1296\*b2 + 8772\*c2 = 319,1

Resolvendo o sistema, os valores de a2, b2 e c2 são : a2 = 0,407 ; b2 = 0,077; c2 = 0,015

Sendo assim, **φ(x) = 0,407 + 0,077x + 0,015x²**

1. Para a análise entre as duas curvas encontradas, deseja-se descobrir qual é aquela que melhor satisfaz os valores dos pontos apresentados na tabela, sendo feita através do erro quadrático em cada ponto. O cálculo é feito com base na equação a seguir:

*Análise do erro por curva*

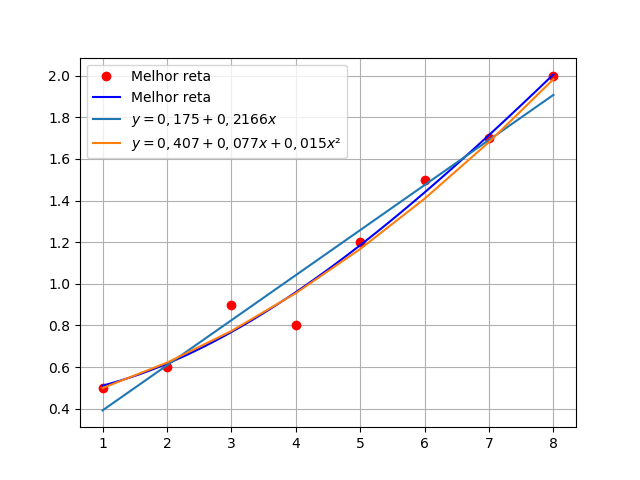
* ***Reta***

Erroreta = 0,391 - 0,082 + 0,0752 - 0,2414 - 0,058 + 0,0254 + 0,0088 + 0,0922 = 0,1190

* ***Parábola***

Erroparabola = 0,0001 - 0,021 + 0,127 - 0,155 + 0,033 + 0,091 + 0,019 + 0,017 = 0,1111

Logo, fazendo a comparação, como Erroreta > Erroparabola, a curva da parábola se aproxima mais dos pontos fornecidos pela tabela. Além disso, ao realizar uma análise visual, tendo a programação em Python como ferramenta, é possível ter o seguinte resultado ao plotar um gráfico com os valores citados acima:



### **Exercício 2**

Para a resolução deste exercício, a mesma metodologia utilizada acima será usada, porém, com valores diferentes de x e f(x). Tem como g1(x), g2(x) e g3(x) o seguintes valores:

* g1(x) = 1
* g2(x) = x
* g3(x) = x²

Os valores das constantes a2, b2 e c2, de modo que a função φ(x) = a2\*g1(x) + b2\*g2(x) + c2\*g3(x) se aproxime ao máximo de f(x) é dado por:

a2 ++=

a2 ++=

a2 ++=

Assim, tem-se o seguinte sistema:

11\*a2 + 21369\*b2 + 41.53\*10^6\*c2 = 750.9\*10^6

21369\*a2 + 41.53\*10^6\*b2 + 8.096\*10^10\*c2 = 1.4798\*10^12

41.53\*10^6\*a2 + 8.096\*10^10\*b2 +1.571\*10^14\*c2 = 2.9167\*10^15

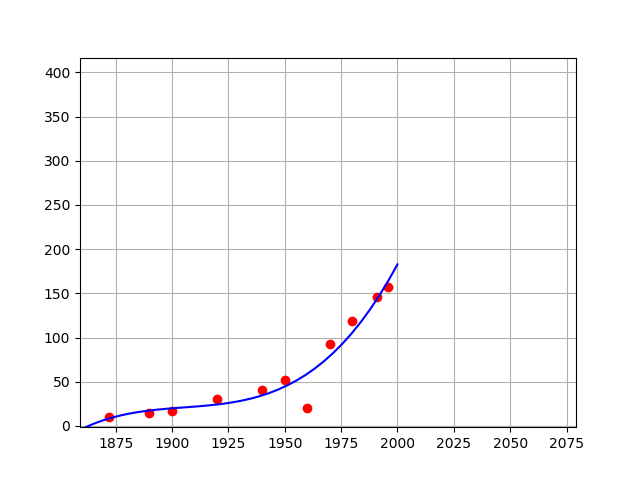
Utilizando a **programação em Python** como ferramenta para resolução deste sistema, obteve-se os seguintes valores de a2, b2 e c2 : a2 = -5.95332131e+08 ; b2 = 2.23966770e+05; c2 = 6.05251034e+01

Sendo assim, **φ(x) = -5.95332131e+08 + 2.23966770e+05\*x + 6.05251034e+01\*x²**

Logo, no ano de 2000, segundo a curva modelada teria uma população igual a **94701822.6** .

### População brasileira real no ano de 2000 = **169.799.170**

Gráfico da curva modelada

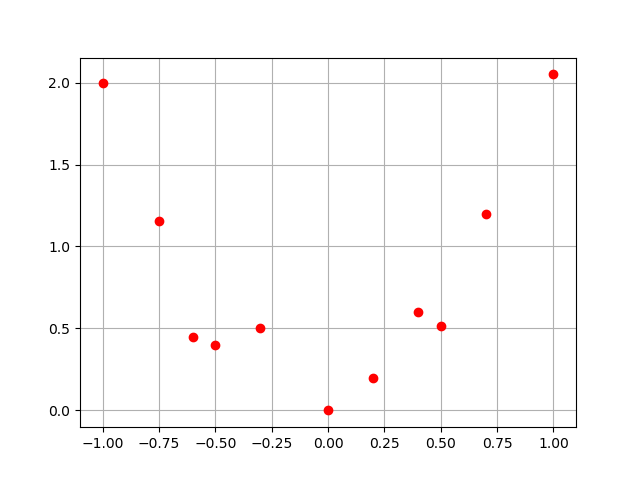


### **Exercício 3**

A tabela 1 foi a proposta pelo exercício para plotar um gráfico a fim de observar o comportamento do conjunto de pontos. Logo, a tabela e o gráfico apenas com os pontos são:

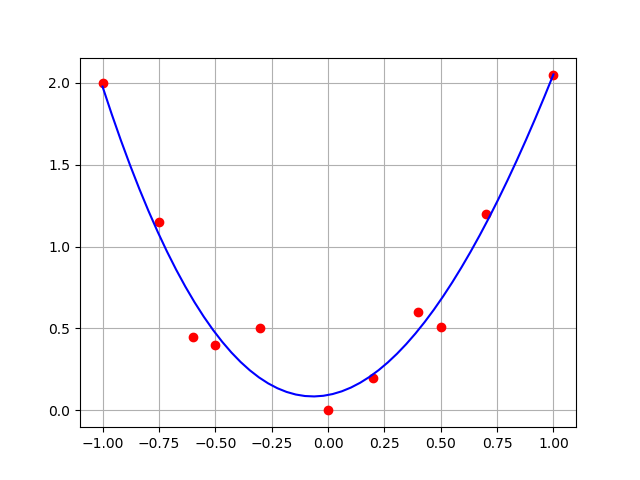
**Tabela 1**: Relação de Números

| **x** | -1.0 | -0.75 | -0.6 | -0.5 | -0.3 | 0.0 | 0.2 | 0.4 | 0.5 | 0.7 | 1.0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **f(x)** | 2.0 | 1.153 | 0.45 | 0.4 | 0.5 | 0.0 | 0.2 | 0.6 | 0.512 | 1.2 | 2.05 |



Ao plotar o gráfico de acordo com a tabela 1 foi possível perceber que o conjunto se comporta aproximadamente como uma parábola, ou seja, uma equação da forma (polinômio de segundo grau) com o coeficiente A maior que zero devido a concavidade para cima. Pode-se notar que, neste caso em específico, a resolução por fórmula de Bhaskara gera um delta igual a zero, mostrando o ponto (0,0) como o ponto mais baixo da curva o qual tangencia o eixo x(expõe que há apenas uma raiz real como solução da equação).

Ao aplicar o **Método dos Mínimos Quadrados** foi possível notar que uma parábola de fato era a curva que mais se aproximava das exigências propostas pelo método(explicitadas no Exercício 1 e aplicadas neste exercício). Assim o gráfico gerado foi:



Usando o **programa** em Anexo para aplicar os dados apresentados às condições para o Método dos Mínimos Quadrados foi possível encontrar o polinômio ideal para os pontos. Assim, a **equação** que modela a parábola é .

### **Exercício 4**

Neste exercício, o objetivo é transformar uma equação não linear em linear a fim de podermos aplicar o **Método dos Mínimos Quadrados**. Assim, usando apenas as primeiras colunas de R e W, tomemos os pontos:

| **W** | **R** |
| --- | --- |
| 0.017 | 0.154 |
| 0.087 | 0.296 |
| 0.174 | 0.363 |
| 1.11 | 0.531 |
| 1.74 | 2.23 |
| 4.09 | 3.58 |
| 5.45 | 3.52 |
| 5.96 | 2.40 |

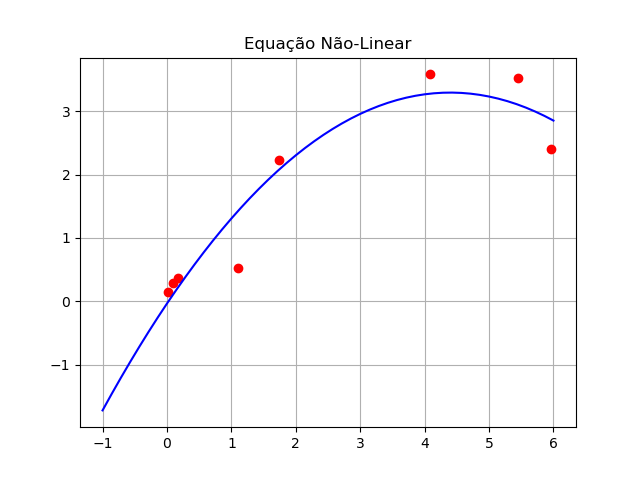
A ideia é aplicar esses pontos à equação (equação 1), mas como a equação **não é linear**, primeiro temos que torná-la adequada. Dessa forma, partindo da equação (1):

As equações (3) e (4) geram um novo conjunto de pontos, sendo eles:

Substituindo as equações (3), (4) e (5) em (2):

Assim, chegamos a equação (5), a qual por ser linear pode ser usada para o Método dos Mínimos Quadrados. Resolvendo a equação com o auxílio da programação em Python, podemos chegar ao valor de b e e substituir na equação (1).

De acordo com o cálculo feito em Python, o valor alfa foi de **0.5462634**. Logo, podemos utilizar o Método dos Mínimos Quadrados para verificar a equação mais adequada uma vez que ela se encaixa como um **polinômio de primeiro grau**(). O gráfico da curva com os pontos e a curva adequada resultou na imagem abaixo:



# **Anexo Parte 1**

**Exercicio 1:**

import numpy as np

import numpy as poly

x = np.array([5, 10, 15, 20, 30, 40], dtype="double")

y = np.array([1226, 1498, 1822, 2138, 2662, 2840], dtype="double")

#inicializando a tabela

T = np.zeros((6,6));

#primeira coluna

T[:,0]=y;

#segunda coluna

T[1,1]=(T[1,0]-T[0,0])/(x[1]-x[0]);

T[2,1]=(T[2,0]-T[1,0])/(x[2]-x[1]);

T[3,1]=(T[3,0]-T[2,0])/(x[3]-x[2]);

T[4,1]=(T[4,0]-T[3,0])/(x[4]-x[3]);

T[5,1]=(T[5,0]-T[4,0])/(x[5]-x[4]);

#terceira coluna

T[2,2]=(T[2,1]-T[1,1])/(x[2]-x[0]);

T[3,2]=(T[3,1]-T[2,1])/(x[3]-x[1]);

T[4,2]=(T[4,1]-T[3,1])/(x[4]-x[2]);

T[5,2]=(T[5,1]-T[4,1])/(x[5]-x[3]);

#quarta coluna

T[3,3]=(T[3,2]-T[2,2])/(x[3]-x[0]);

T[4,3]=(T[4,2]-T[3,2])/(x[4]-x[1]);

T[5,3]=(T[5,2]-T[4,2])/(x[5]-x[2]);

#quinta coluna

T[4,4]=(T[4,3]-T[3,3])/(x[4]-x[0]);

T[5,4]=(T[5,3]-T[4,3])/(x[5]-x[1]);

#sexta coluna

T[5,5]=(T[5,4]-T[4,4])/(x[5]-x[0]);

#polinomio interpolador

p = np.array([T[0,0]], dtype="double")

paux = np.array([-x[0],1], dtype="double")

p.resize(2)

p += T[1,1]\*paux

paux = poly.polymul(paux,[-x[1],1])

p.resize(3)

p += T[2,2]\*paux

paux = poly.polymul(paux,[-x[2],1])

p.resize(4)

p += T[3,3]\*paux

paux = poly.polymul(paux,[-x[3],1])

p.resize(5)

p += T[4,4]\*paux

paux = poly.polymul(paux,[-x[4],1])

p.resize(6)

p += T[5,5]\*paux

x = 12

x1 = p[0] + x\*p[1] + (x \*\* 2)\*p[2] + (x \*\* 3)\*p[3] + (x \*\* 4)\*p[4] + (x \*\* 5)\*p[5]

print("O valor do ponto", x, "eh:", x1)

x = 25

x1 = p[0] + x\*p[1] + (x \*\* 2)\*p[2] + (x \*\* 3)\*p[3] + (x \*\* 4)\*p[4] + (x \*\* 5)\*p[5]

print("O valor do ponto", x, "eh:", x1)

x = 38

x1 = p[0] + x\*p[1] + (x \*\* 2)\*p[2] + (x \*\* 3)\*p[3] + (x \*\* 4)\*p[4] + (x \*\* 5)\*p[5]

print("O valor do ponto", x, "eh:", x1)

**Exercicio 2:**

**import numpy as np**

**import numpy as poly**

**x = np.array([1, 2, 3, 4, 5, 6], dtype="double")**

**y = np.array([14.5, 19.5, 30.5, 53.5, 94.5, 159.5], dtype="double")**

**#inicializando a tabela**

**T = np.zeros((6,6));**

**#primeira coluna**

**T[:,0]=y;**

**#segunda coluna**

**T[1,1]=(T[1,0]-T[0,0])/(x[1]-x[0]);**

**T[2,1]=(T[2,0]-T[1,0])/(x[2]-x[1]);**

**T[3,1]=(T[3,0]-T[2,0])/(x[3]-x[2]);**

**T[4,1]=(T[4,0]-T[3,0])/(x[4]-x[3]);**

**T[5,1]=(T[5,0]-T[4,0])/(x[5]-x[4]);**

**#terceira coluna**

**T[2,2]=(T[2,1]-T[1,1])/(x[2]-x[0]);**

**T[3,2]=(T[3,1]-T[2,1])/(x[3]-x[1]);**

**T[4,2]=(T[4,1]-T[3,1])/(x[4]-x[2]);**

**T[5,2]=(T[5,1]-T[4,1])/(x[5]-x[3]);**

**#quarta coluna**

**T[3,3]=(T[3,2]-T[2,2])/(x[3]-x[0]);**

**T[4,3]=(T[4,2]-T[3,2])/(x[4]-x[1]);**

**T[5,3]=(T[5,2]-T[4,2])/(x[5]-x[2]);**

**#quinta coluna**

**T[4,4]=(T[4,3]-T[3,3])/(x[4]-x[0]);**

**T[5,4]=(T[5,3]-T[4,3])/(x[5]-x[1]);**

**#sexta coluna**

**T[5,5]=(T[5,4]-T[4,4])/(x[5]-x[0]);**

**#polinomio interpolador**

**p = np.array([T[0,0]], dtype="double")**

**paux = np.array([-x[0],1], dtype="double")**

**p.resize(2)**

**p += T[1,1]\*paux**

**paux = poly.polymul(paux,[-x[1],1])**

**p.resize(3)**

**p += T[2,2]\*paux**

**paux = poly.polymul(paux,[-x[2],1])**

**p.resize(4)**

**p += T[3,3]\*paux**

**paux = poly.polymul(paux,[-x[3],1])**

**p.resize(5)**

**p += T[4,4]\*paux**

**paux = poly.polymul(paux,[-x[4],1])**

**p.resize(6)**

**p += T[5,5]\*paux**

**x = 4.5**

**x1 = p[0] + x\*p[1] + (x \*\* 2)\*p[2] + (x \*\* 3)\*p[3] + (x \*\* 4)\*p[4] + (x \*\* 5)\*p[5]**

**print("O valor do ponto", x, "eh:", x1)**

**Exercício 3:**

**import numpy as np**

**import numpy as poly**

**x = np.array([2, 4.25], dtype="double")**

**y = np.array([7.2, 7.1], dtype="double")**

**#inicializando a tabela**

**T = np.zeros((2,2));**

**#primeira coluna**

**T[:,0]=y;**

**#segunda coluna**

**T[1,1]=(T[1,0]-T[0,0])/(x[1]-x[0]);**

**#polinomio interpolador**

**p = np.array([T[0,0]], dtype="double")**

**paux = np.array([-x[0],1], dtype="double")**

**p.resize(2)**

**p += T[1,1]\*paux**

**paux = poly.polymul(paux,[-x[1],1])**

**x = 4**

**x1 = p[0] + x\*p[1]**

**print("O valor do ponto", x, "eh:", x1)**

# **Anexo Parte 2**

## **Exercício 1:**

### **a)**

**from \_\_future\_\_ import division**

**import numpy as np**

**from numpy import linalg**

**import matplotlib.pyplot as plt**

**xi = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0], dtype = 'double')**

**yi = np.array([0.5, 0.6, 0.9, 0.8, 1.2, 1.5, 1.7, 2.0], dtype = 'double')**

**y = 0.175 + 0.2166\*xi**

**y2 = 0.407 + 0.077\*xi + 0.015\*xi\*\*2**

**V = np.array([xi\*\*3,xi\*\*2,xi\*\*1, xi\*\*0]).transpose() #mostra qual a melhor funcao para o caso sendo polinomio**

**a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi)**

**xx = np.linspace(1.0,8.0) #intervalo do grafico**

**plt.plot(xi,yi, 'ro',xx,np.polyval(a,xx),'b-', label = 'Melhor reta'.format()) # MELHOR FUNCAO**

**plt.plot(xi, y, label = '$y = 0,175 + 0,2166x$'.format()) # RETA**

**plt.plot(xi, y2, label = '$y = 0,407 + 0,077x + 0,015x²$'.format()) #PARABOLA**

**plt.plot(())**

**plt.grid()**

**plt.legend(loc='best')**

**plt.show()**

### **b)** Cálculo sistema

**import numpy as np**

**a = np.array([[11.0, 21369.0, 41.53\*10\*\*6], [21369.0, 41.53\*10\*\*6, 8.096\*10\*\*10], [41.53\*10\*\*6, 8.096\*10\*\*10, 1.571\*10\*\*14]])**

**b = np.array([750.9\*10\*\*6, 1.4798\*10\*\*12, 2.9167\*10\*\*15])**

**x = np.linalg.solve(a, b)**

**print(x)**

**Cálculo curva com base nos pontos e gráfico**

**from \_\_future\_\_ import division**

**import numpy as np**

**from numpy import linalg**

**import matplotlib.pyplot as plt**

**xi = np.array([1872.0, 1890.0, 1900.0, 1920.0, 1940.0, 1950.0, 1960.0, 1970.0, 1980.0, 1991.0, 1996.0], dtype = 'double')**

**yi = np.array([9.9, 14.3, 17.4, 30.6, 41.2, 51.9, 20.2, 93.1, 119.0, 146.2, 157.1], dtype = 'double')**

**V = np.array([xi\*\*3,xi\*\*2,xi\*\*1, xi\*\*0]).transpose() #mostra qual a melhor funcao para o caso sendo polinomio**

**a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi)**

**xx = np.linspace(1800.0, 2000.0) #intervalo do grafico**

**plt.plot(xi,yi, 'ro',xx,np.polyval(a,xx),'b-') # MELHOR FUNCAO**

**plt.plot(())**

**plt.grid()**

**plt.show()**

## **Exercício 3**

* **Gráfico para mostrar os pontos mostrados em tabela pelo enunciado**

**from \_\_future\_\_ import division**

**import numpy as np**

**from numpy import linalg**

**import matplotlib.pyplot as plt**

**xi = np.array([-1.0, -0.75, -0.6, -0.5, -0.3, 0.0, 0.2, 0.4, 0.5, 0.7, 1], dtype = 'double')**

**yi = np.array([2.0, 1.153, 0.45, 0.4, 0.5, 0.0, 0.2, 0.6, 0.512, 1.2, 2.05], dtype = 'double')**

**V = np.array([xi\*\*1,xi\*\*0]).transpose()**

**xx = np.linspace(0.5,2.5)**

**plt.plot(xi,yi, 'ro')**

**plt.grid()**

**plt.show()**

* Programa que gera a melhor curva adequando os pontos

from \_\_future\_\_ import division

import numpy as np

from numpy import linalg

import matplotlib.pyplot as plt

xi = np.array([-1.0, -0.75, -0.6, -0.5, -0.3, 0.0, 0.2, 0.4, 0.5, 0.7, 1.0], dtype = 'double')

yi = np.array([2.0, 1.153, 0.45, 0.4, 0.5, 0.0, 0.2, 0.6, 0.512, 1.2, 2.05], dtype = 'double')

V = np.array([xi\*\*2,xi\*\*1, xi\*\*0]).transpose() #mostra qual a melhor funcao para o caso sendo polinomio

a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi)

xx = np.linspace(-1.0,1.0) #intervalo do grafico

plt.plot(xi,yi,'ro',xx,np.polyval(a,xx),'b-')

plt.grid()

**plt.show()**

## **Exercício 4:**

* Programa que adapta a equação não linear e aplica o Método dos Mínimos Quadrados

**from \_\_future\_\_ import division**

**import numpy as np**

**from numpy import linalg**

**import matplotlib.pyplot as plt**

**W = np.array([0.017, 0.087, 0.174, 1.11, 1.74, 4.09, 5.45, 5.96])**

**R = np.array([0.154, 0.296, 0.363, 0.531, 2.23, 3.58, 3.52, 2.40])**

**V = np.array([np.ones(8), np.log(W)]).transpose()**

**a = np.linalg.lstsq(V, np.log(R))[0]**

**b = W\*\*a[0]**

**alfa = a[1]**

**reg = b\*(W\*\*alfa)**

**curva = np.array([W\*\*2,W\*\*1, W\*\*0]).transpose()**

**metodo = ((np.linalg.inv((curva.transpose()).dot(curva))).dot(curva.transpose())).dot(R)**

**xx = np.linspace(-1.0, 6.0)**

**plt.plot(W,R,'ro',xx,np.polyval(metodo,xx), 'b-')**

**plt.title('Equação Não-Linear')**

**plt.grid()**

**plt.show()**