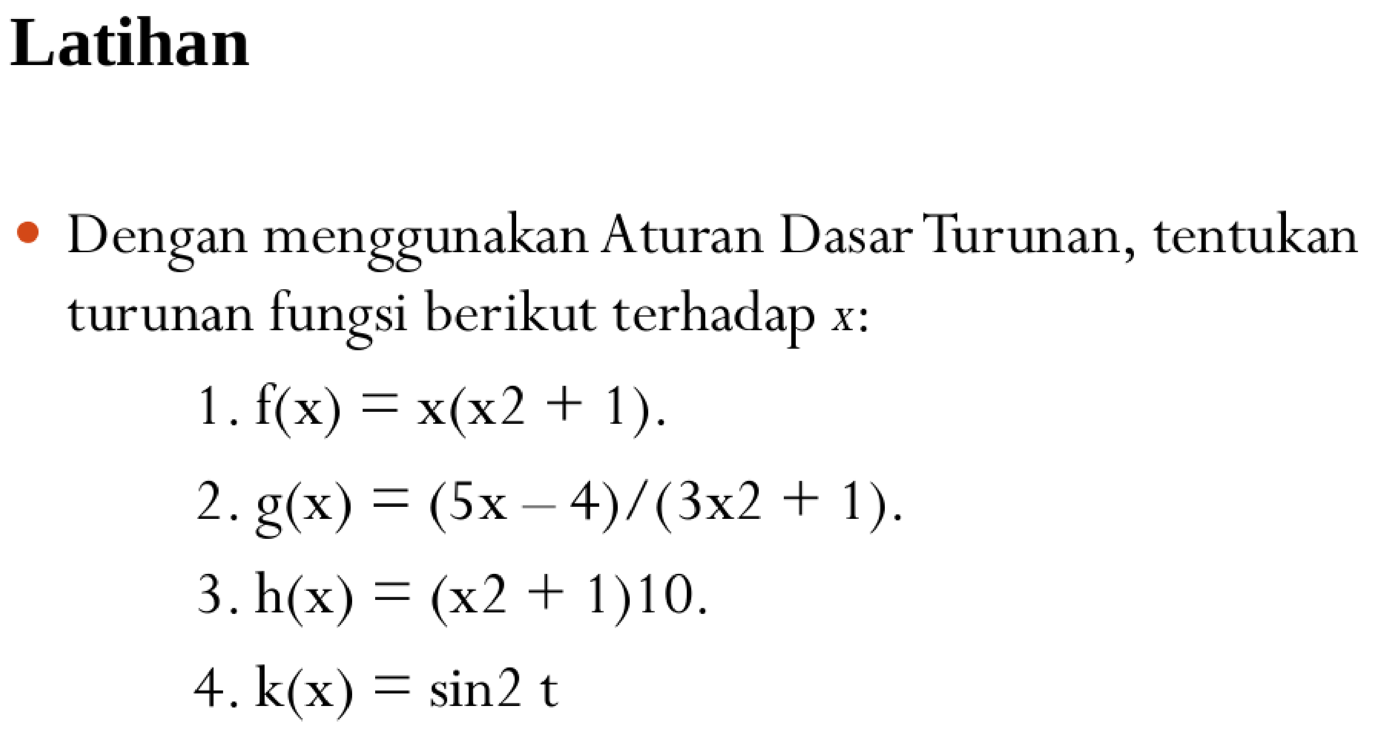
Nama : Vika Putri Ariyanti

Kelas : 4IA88

NPM : 56417094

TUGAS 1

Turunan



Latihan

- Dengan menggunakan Aturan Dasar Turunan, tentukan turunan fungsi berikut terhadap *x*:

1. f(x) = x(x2 + 1).
2. g(x) = (5x – 4)/(3x2 + 1).
3. h(x) = (x2 + 1)10.
4. k(x) = sin2 t

In [1]:

```
from sympy import *
import sympy as sp
sp.init_printing()

x = sp.symbols('x')
t = sp.symbols('t')
f = sp.Function('f')(x)
f1 = sp.Function("f\'")(x)
g = sp.Function('g')(x)
g1 = sp.Function("g\'")(x)
h = sp.Function('h')(x)
h1 = sp.Function("h\'")(x)
k = sp.Function('k')(x)
k1 = sp.Function("k\'")(x)
```

In [2]:

```
nomer1 = x*(x**2+1)
soal1 = sp.Eq(f, nomer1)
display(soal1)
jawaban = nomer1.diff(x)
jawaban1 = sp.Eq(f1, jawaban)
display(jawaban1)
```

$$f(x) = x\left(x^2 + 1\right)$$

$$\mathrm{f'}(x) = 3x^2 + 1$$

In [3]:

```
nomer2 = (5*x-4)/(3*x**2+1)
soal2 = sp.Eq(g, nomer2)
display(soal2)
jawaban = nomer2.diff(x)
jawaban2 = sp.Eq(g1, jawaban)
display(jawaban2)
```

$$g(x) = \frac{5x - 4}{3x^2 + 1}$$

$$\mathrm{g}'(x) = -\frac{6x(5x - 4)}{(3x^2 + 1)^2} + \frac{5}{3x^2 + 1}$$

In [4]:

```
nomer3 = (x**2+1)*10
soal3 = sp.Eq(h, nomer3)
display(soal3)
jawaban = nomer3.diff(x)
jawaban3 = sp.Eq(h1, jawaban)
display(jawaban3)
```

$$h(x) = 10x^2 + 10$$

$$\mathrm{h}'(x) = 20x$$

In [5]:

```
nomer4 = sin(2*t)
soal4 = sp.Eq(k, nomer4)
display(soal4)
jawaban = nomer4.diff(x)
jawaban4 = sp.Eq(k1, jawaban)
display(jawaban4)
```

$$k(x) = \sin(2t)$$

$$\mathrm{k}'(x) = 0$$

Nilai Signifikan

In [6]:

```
r = 1
phi = [3.1, 3.14, 3.141, 3.1415, 3.14159,
       3.141592, 3.1415926, 3.14159265, 3.141592653, 3.1415926535,
       3.14159265358, 3.141592653589, 3.1415926535893, 3.14159265358932, 3.141592653589323,
       3.1415926535893238, 3.1415926535893284, 3.14159265358932846]
phii = map(str, phi)
```

In [7]:

```
AS = []
for i in phi:
  AS.append(len(str(i))-1)
AS
```

Out[7]:

$$[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 17, 17]$$

In [8]:

```
luas = phi * (r ** 2)
luass = [str(x) for x in luas]
AS_luas = []
for i in luas:
  AS_luas.append(len(str(i))-1)
AS_luas
```

Out[8]:

$$[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 17, 17]$$

In [9]:

```
volume = [(4/3) * x for x in phi]
volumee = [str(x) for x in volume]
AS_volume = []
for i in volume:
  AS_volume.append(len(str(i))-1)
AS_volume
```

Out[9]:

[16, 17, 4, 16, 16, 16, 16, 8, 10, 16, 16, 16, 16, 15, 17, 16, 17, 17]

In [10]:

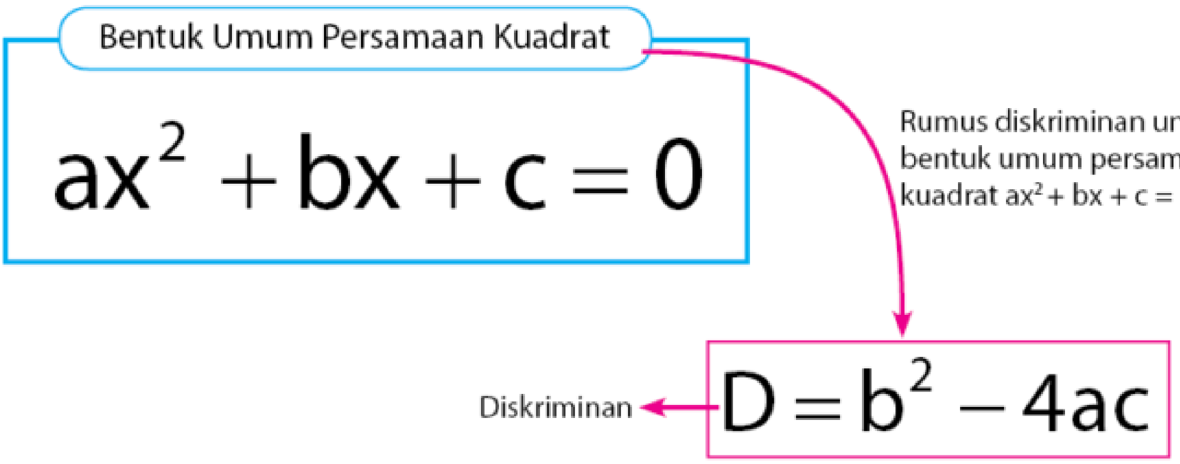
```
import pandas as pd
dict = {'Nilai Phi': phii, 'Nilai Signifikan':AS,'Luas Lingkaran': luass, 'Angka Signifikan Luas Lingkaran': AS_luas,
        'Volume Bola': volumee, 'Angka Signifikan Volume Bola': AS_volume}
df = pd.DataFrame(dict)
df
```

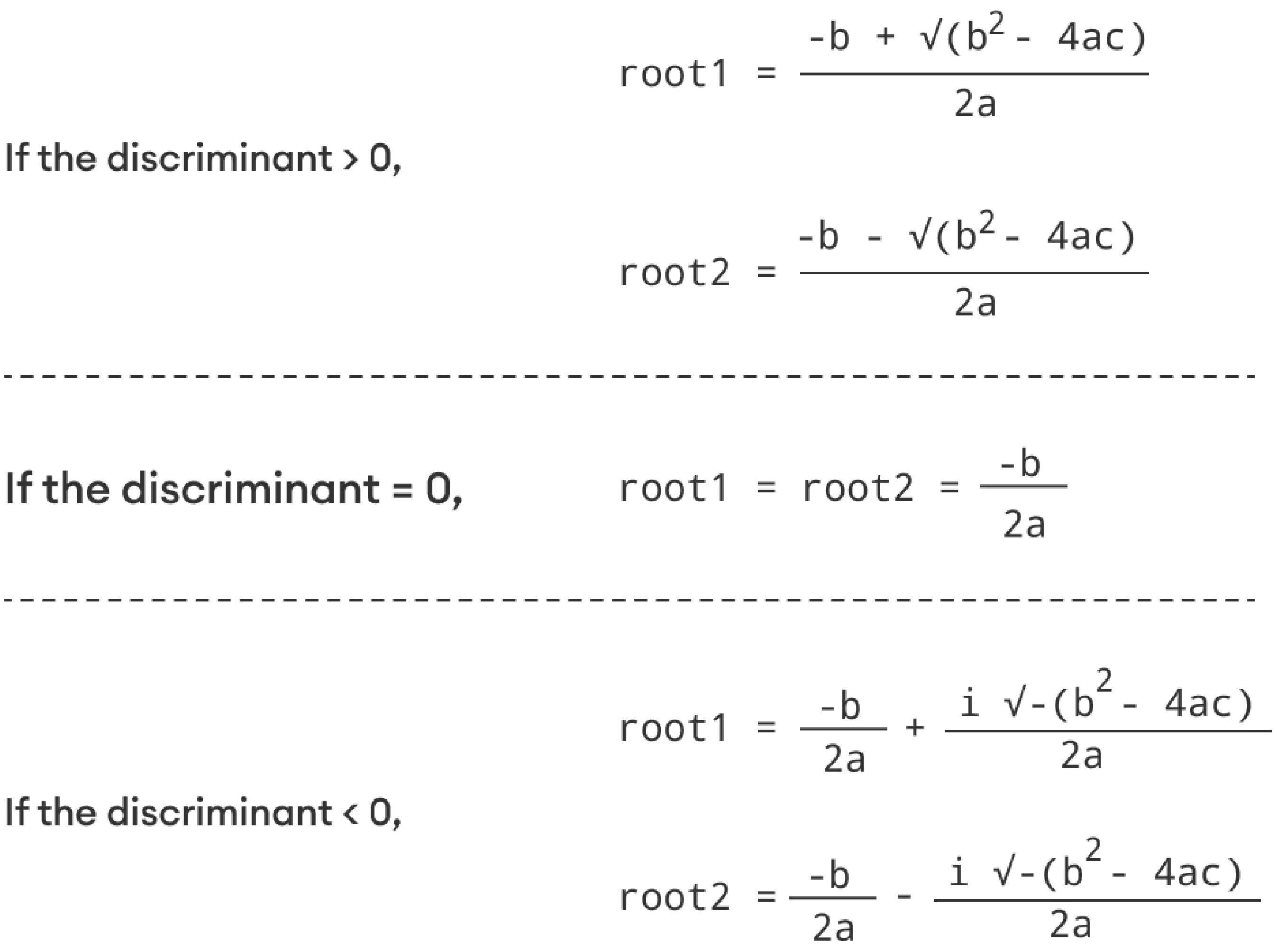
Out[10]:

	Nilai Phi	Nilai Signifikan	Luas Lingkaran	Angka Signifikan Luas Lingkaran	Volume Bola	Angka Signifikan Volume Bola
0	3.1	2	3.1	2	4.133333333333333	16
1	3.14	3	3.14	3	4.1866666666666665	17
2	3.141	4	3.141	4	4.188	4
3	3.1415	5	3.1415	5	4.188666666666666	16
4	3.14159	6	3.14159	6	4.188786666666666	16
5	3.141592	7	3.141592	7	4.188789333333333	16
6	3.1415926	8	3.1415926	8	4.188790133333333	16
7	3.14159265	9	3.14159265	9	4.1887902	8
8	3.141592653	10	3.141592653	10	4.188790204	10
9	3.1415926535	11	3.1415926535	11	4.188790204666667	16
10	3.14159265358	12	3.14159265358	12	4.188790204773333	16
11	3.141592653589	13	3.141592653589	13	4.188790204785333	16
12	3.1415926535893	14	3.1415926535893	14	4.188790204785733	16
13	3.14159265358932	15	3.14159265358932	15	4.18879020478576	15
14	3.141592653589323	16	3.141592653589323	16	4.1887902047857635	17
15	3.1415926535893237	17	3.1415926535893237	17	4.188790204785764	16
16	3.1415926535893286	17	3.1415926535893286	17	4.1887902047857715	17
17	3.1415926535893286	17	3.1415926535893286	17	4.1887902047857715	17

TUGAS 2

PERSAMAAN KUADRAT


Bentuk Umum Persamaan Kuadrat
$ax^2 + bx + c = 0$
Rumus diskriminan untuk bentuk umum persamaan kuadrat $ax^2 + bx + c = 0$
Diskriminan
$D = b^2 - 4ac$



If the discriminant > 0,

$$\text{root1} = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$\text{root2} = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

If the discriminant = 0,

$$\text{root1} = \text{root2} = \frac{-b}{2a}$$

If the discriminant < 0,

$$\text{root1} = \frac{-b}{2a} + \frac{i\sqrt{-(b^2 - 4ac)}}{2a}$$

$$\text{root2} = \frac{-b}{2a} - \frac{i\sqrt{-(b^2 - 4ac)}}{2a}$$

Menghitung Nilai d

In [11]:

```
import math

def akarkuadrat(a, b, c):
    d = b * b - 4 * a * c
    akar = math.sqrt(abs(d))

    if d > 0:
        print('akar nyata dan berbeda, yaitu', (-b + akar)/(2 * a)), 'dan', ((-b - akar)/(2
* a))

    elif d == 0:
        print('akar nyata dan sama, yaitu', (-b)/(2 * a))

    else:
        print("Akar Kompleks, yaitu")
        print(- b / (2 * a), akar, " + i")
        print(- b / (2 * a), akar, " - i")
```

Menemukan Solusi

In [12]:

```
a = float(input('Masukkan nilai a: '))
b = float(input('Masukkan nilai b: '))
c = float(input('Masukkan nilai c: '))
print('')

if a == 0:
        print("Masukkan persamaan kuadrat yang benar")

else:
    akarkuadrat(a, b, c)
```

```
Masukkan nilai a: 1
Masukkan nilai b: 10
Masukkan nilai c: 25

akar nyata dan sama, yaitu -5.0
```

Persamaan Kubik

1. Metode Biseksi

In [13]:

```
def bisection(f,a,b,N):
    if f(a)*f(b) >= 0:
        print("Metode bisection gagal.")
        return None
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = (a_n + b_n)/2
        f_m_n = f(m_n)
        if f(a_n)*f_m_n < 0:
            a_n = a_n
            b_n = m_n
        elif f(b_n)*f_m_n < 0:
            a_n = m_n
            b_n = b_n
        elif f_m_n == 0:
            print("Menemukan solusi yang tepat.")
            return m_n
        else:
            print("Metode bisection gagal.")
            return None
    return (a_n + b_n)/2
```

In [14]:

```
f = lambda x: x**2 - x - 1
hasil = bisection(f,1,2,25)
print(hasil)
```

```
1.618033990263939
```

2. Metode Newton-Raphson

In [15]:

```
def newton(f,Df,x0,epsilon,max_iter):
    xn = x0
    for n in range(0,max_iter):
        fxn = f(xn)
        if abs(fxn) < epsilon:
            print('Solusi ditemukan setelah',n,'iterasi.')
            return xn
        Dfxn = Df(xn)
        if Dfxn == 0:
            print('Turunan nol. Tidak ada solusi yang ditemukan.')
            return None
        xn = xn - fxn/Dfxn
    print('Melebihi iterasi maksimum. Tidak ada solusi yang ditemukan.')
    return None
```

In [16]:

```
f = lambda x: x**2 - x - 1
Df = lambda x: 2*x - 1
newton(f,Df,1,1e-8,10)
```

```
Solusi ditemukan setelah 5 iterasi.
```

Out[16]:

```
1.618033988749989
```

3. Metode Sekan

In [17]:

```
def sekan(f,a,b,N):
    if f(a)*f(b) >= 0:
        print("Metode Sekan Gagal.")
        return None
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
        f_m_n = f(m_n)
        if f(a_n)*f_m_n < 0:
            a_n = a_n
            b_n = m_n
        elif f(b_n)*f_m_n < 0:
            a_n = m_n
            b_n = b_n
        elif f_m_n == 0:
            print("Menemukan solusi yang tepat.")
            return m_n
        else:
            print("Metode Sekan Gagal.")
            return None
    return a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
```

In [18]:

```
p = lambda x: x**3 - x**2 - 1
hasil = sekan(p,1,2,20)
print(hasil)
```

```
1.4655712311394433
```

TUGAS 3

Penjumlahan Matriks

In [19]:

```
import numpy as np

A = np.array([[2, 3, 5], [4, 1, 2]])
B = np.array([[1, 2, 6], [3, 2, 1]])
C = A + B
print(C)
```

```
[[ 3  5 11]
 [ 7  3  3]]
```

Pengurangan Matriks

In [20]:

```
A = np.array([[2, 6], [-4, 1], [3, 2]])
B = np.array([[6, -2], [4, 1], [0, 3]])
C = A - B
print(C)
```

```
[[-4  8]
 [-8  0]
 [ 3 -1]]
```

Perkalian Skalar pada Matriks

In [21]:

```
A = np.array([[2, 6], [-4, 1], [3, 2]])
C = 2 * A
print(C)
```

```
[[ 4 12]
 [-8  2]
 [ 6  4]]
```

Perkalian Matriks dengan Matriks

In [22]:

```
A = np.array([[2, 1, -6], [1, -3, 2]])
B = np.array([[1, 0, -3], [0, 4, 2], [-2, 1, 1]])
C = A.dot(B)
print(C)
```

```
[[ 14  -2 -10]
 [ -3 -10  -7]]
```

Determinan

In [23]:

```
A = np.array([[3, -7], [-9, 5]])
determinan = np.linalg.det(A)
print(round(determinan))
```

```
-48
```

In [24]:

```
A = np.array([[3, 0, -2], [6, -8, 1], [0, 3, 4]])
determinan = np.linalg.det(A)
print(round(determinan))
```

```
-141
```

Invers

In [25]:

```
A = np.array([[1, 2, 3], [2, 4, 3], [6, 1, 4]])
invers = np.linalg.inv(A)
print(invers)
```

```
[[-0.39393939  0.15151515  0.18181818]
 [-0.3030303   0.42424242 -0.09090909]
 [ 0.66666667 -0.33333333  0.        ]]
```

Metode Eliminasi Gauss

In [26]:

```
import sys

n = int(input('Masukkan jumlah variabel yang tidak diketahui: '))
a = np.zeros((n,n+1))
x = np.zeros(n)
```

```
Masukkan jumlah variabel yang tidak diketahui: 2
```

In [27]:

```
print('Masukkan Koefisien Augmented Matriks:')
for i in range(n):
    for j in range(n+1):
        a[i][j] = float(input( 'a['+str(i)+']['+ str(j)+']='))

for i in range(n):
    if a[i][i] == 0.0:
        sys.exit('Terdekeksi dibagi dengan nol')

    for j in range(i+1, n):
        ratio = a[j][i]/a[i][i]

        for k in range(n+1):
            a[j][k] = a[j][k] - ratio * a[i][k]
```

```
Masukkan Koefisien Augmented Matriks:
a[0][0]=1
a[0][1]=2
a[0][2]=3
a[1][0]=4
a[1][1]=5
a[1][2]=6
```

Proses Substitusi Mundur

In [28]:

```
x[n-1] = a[n-1][n]/a[n-1][n-1]

for i in range(n-2,-1,-1):
    x[i] = a[i][n]

    for j in range(i+1,n):
        x[i] = x[i] - a[i][j]*x[j]

    x[i] = x[i]/a[i][i]
```

In [29]:

```
print('\nSolusinya adalah: ')
```

```
for i in range(n):
    print('X%d = %f' %(i,x[i]), end = '\t')
```

```
Solusinya adalah:
X0 = -1.000000 X1 = 2.000000
```

Metode Eliminasi Gauss Seidell

inisialisasi matrik A

In [30]:

```
from numpy import array,zeros

A = array([[10.,-1.,2.,0.],
           [-1.,11.,-1.,3.],
           [2.,-1.,10.,-1.],
           [0.,3.,-1.,8.]])
print(A)
```

```
[[10. -1.  2.  0.]
 [-1. 11. -1.  3.]
 [ 2. -1. 10. -1.]
 [ 0.  3. -1.  8.]]
```

inisialisasi vektor b

In [31]:

```
b = array([[6.],
           [25],
           [-11],
           [15]])
print(b)
```

```
[[  6.]
 [ 25.]
 [-11.]
 [ 15.]]
```

In [32]:

```
n=len(A)
iterasi=500
toleransi=0.0001
xlama=zeros((n,1))
xbaru=zeros((n,1))
c=zeros((n,1))
T=zeros((n,n))
```

Menghitung matrik T dan vektor c

In [33]:

```
for j in range(0,n-1):
  for i in range(0,n):
    if i==j:
      i=i+1
    T[j][i]=-1.*A[j][i]/A[j][j]
  c[j][0]=b[j][0]/A[j][j]

j=n-1
for i in range(0,n-1):
  T[j][i]=-1.*A[j][i]/A[j][j]
c[j][0]=b[j][0]/A[j][j]
```

Metode Gauss-Seidel

In [34]:

```
for m in range(1,iterasi):
  S=0
  for i in range(0,n):
    S=S+T[0][i]*xlama[i][0]
  xbaru[0][0]=S+c[0][0]

  for k in range(1,n):
    P=0
    for j in range(0,k):
      P=P+T[k][j]*xbaru[j][0]
    S=0
    for i in range(k,n):
      S=S+T[k][i]*xlama[i][0]
    xbaru[k][0]=P+S+c[k][0]

  x=xbaru-xlama

  xlama=xbaru.copy()
```

Mencetak hasil perhitungan

In [35]:

```
print('iterasi ke', m)
print(xbaru)
```

```
iterasi ke 499
[[ 1.]
 [ 2.]
 [-1.]
 [ 1.]]
```

Metode Dekomposisi LU

inisialisasi matrik augment

In [36]:

```
from numpy import array,zeros

A = array([[1, 0, -2, 7],
           [2, -1, 3, 4],
           [3, -3, 1, 5],
           [2, 1, 4, 4]])
print(A)
```

```
[[ 1  0 -2  7]
 [ 2 -1  3  4]
 [ 3 -3  1  5]
 [ 2  1  4  4]]
```

In [37]:

```
b = array([[11],
           [9],
           [8],
           [10]])
print(b)
```

```
[[11]
 [ 9]
 [ 8]
 [10]]
```

In [38]:

```
n = len(A)
L = zeros((n,n))

for i in range(0,n):
  L[i][i]=1
```

Proses Triangularisasi

In [39]:

```
for k in range(0,n-1):
  #pivot
  if A[k][k]==0:
    for s in range(0,n):
      v=A[k][s]
      u=A[k+1][s]
      A[k][s]=u
      A[k+1][s]=v

  for j in range(k+1,n):
    m=A[j][k]/A[k][k]
    L[j][k]=m
    for i in range(0,n):
      A[j][i]=A[j][i]-m*A[k][i]
```

In [40]:

```
U = zeros((n,n))
for i in range(0,n):
  for j in range(0,n):
    U[i][j]=A[i][j]
```

Proses Substitusi Maju

In [41]:

```
y = zeros((n,1))
y[0][0]=b[0][0]/L[0][0]
for j in range(1,n):
  S=0
  for i in range(0,j):
    S=S+y[i][0]*L[j][i]
  y[j][0]=b[j][0]-S
```

Proses Substitusi Mundur

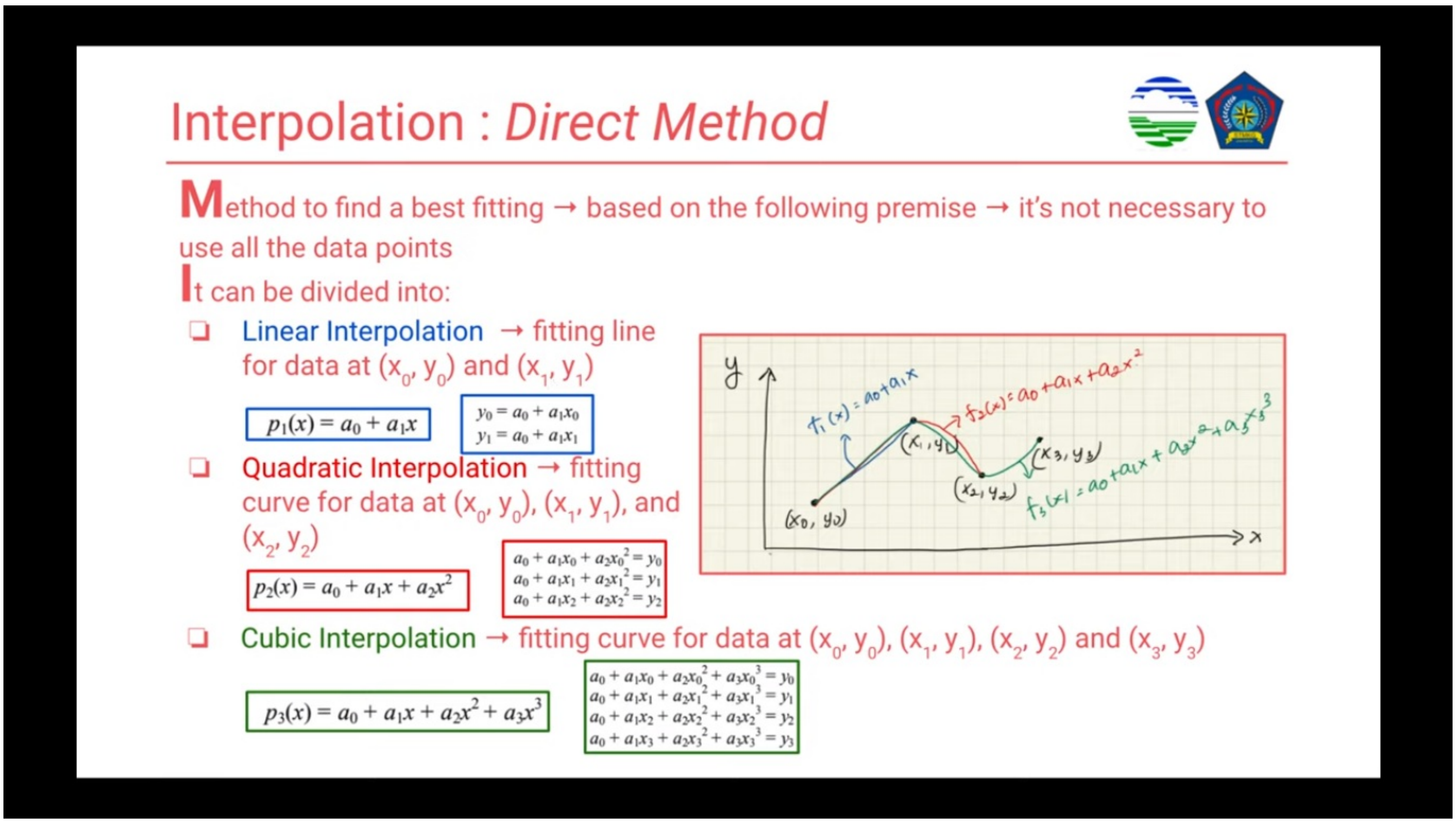
In [42]:

```
x=zeros((n,1))
x[n-1][0]=y[n-1][0]/U[n-1][n-1]

for j in range(n-2,-1,-1):
  S=0
  for i in range(j+1,n):
    S=S+U[j][i]*x[i][0]
  x[j][0]=(y[j][0]-S)/U[j][j]

print(x)
```

```
[[-1.]
 [-0.]
 [ 1.]
 [ 2.]]
```

TUGAS 4

DIRECT METHOD



Interpolation : *Direct Method*

Method to find a best fitting → based on the following premise → it's not necessary to use all the data points

It can be divided into:

- ❑ Linear Interpolation → fitting line for data at (x_0, y_0) and (x_1, y_1)

$$\boxed{p_1(x) = a_0 + a_1x}$$

$$\boxed{\begin{aligned} y_0 &= a_0 + a_1x_0 \\ y_1 &= a_0 + a_1x_1 \end{aligned}}$$

- ❑ Quadratic Interpolation → fitting curve for data at (x_0, y_0), (x_1, y_1), and (x_2, y_2)

$$\boxed{p_2(x) = a_0 + a_1x + a_2x^2}$$

$$\boxed{\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 &= y_1 \\ a_0 + a_1x_2 + a_2x_2^2 &= y_2 \end{aligned}}$$

- ❑ Cubic Interpolation → fitting curve for data at (x_0, y_0), (x_1, y_1), (x_2, y_2) and (x_3, y_3)

$$\boxed{p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3}$$

$$\boxed{\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + a_3x_0^3 &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 &= y_1 \\ a_0 + a_1x_2 + a_2x_2^2 + a_3x_2^3 &= y_2 \\ a_0 + a_1x_3 + a_2x_3^2 + a_3x_3^3 &= y_3 \end{aligned}}$$

In [43]:

```
def terdekat(t, vt, tcari, jml):
    tt = t[:]
    selindex = []
    closestvt = []
    for i in range(jml):
        daftar = []
        for j in range(len(tt)):
            daftar.append(abs(tt[j]-tcari))
        n = daftar.index(min(daftar))
        selindex.append(tt[n])
        tt.remove(tt[n])
    selindex.sort()
    for k in selindex : closestvt.append(vt[t.index(k)])
    return[selindex, closestvt]
```

In [44]:

```
t = [0, 10, 15, 20, 22.5, 30]
vt = [0, 227.04, 362.78, 517.35, 602.97, 901.67]
tcari = 16
points = [2, 3, 4, 5]
vtcari0 = 0
```

In [45]:

```
import numpy
matrixnya = numpy.zeros((len(t), len(t)+1))
for i in range(len(t)):
    matrixnya[i, len(t)] = vt[i]
    for j in range(len(t)):
        matrixnya[i,j] = t[i]**j
```

In [46]:

```
def GaussJordan(A):
    n = len(A)
    x = numpy.zeros(n)
    for k in range(n):
        pivot = A[k][k]
        A[k] = A[k]/pivot
        for i in range(n):
            if i == k: continue
            factor = A[i][k]
            for j in range(k, n+1):
                A[i][j]-factor*A[k][j]
    x = A[:,n]
    return(x)
```

In [47]:

```
def interpdirect(t, vt, tcari):
    matrixnya = numpy.zeros((len(t), len(t)+1))
    for i in range(len(t)):
        matrixnya[i, len(t)] = vt[i]
        for j in range(len(t)):
            matrixnya[i,j]=t[i]**j
    a= GaussJordan(matrixnya)
    vtcari = 0
    for i in range(len(a)):
        vtcari += a[i]*tcari**i
    return[a, vtcari]
```

In [48]:

```
for i in points:
    [tx, vtx] = terdekat(t, vt, tcari, i)
    [a, vtcari] = interpdirect(tx, vtx, tcari)
    print("Orde ", i-1, " Nilai Kecepatan jatuh pada t ke ", tcari, " = ", "%.2f"%vtcari)
    if vtcari0 == 0:
      print("Error = -")
    else:
      print("Error = ", "%.5f"%abs((vtcari-vtcari0)/vtcari*100), "%")
    vtcari0 = vtcari
```

```
Orde  1  Nilai Kecepatan jatuh pada t ke  16  =  776.66
Error = -
Orde  2  Nilai Kecepatan jatuh pada t ke  16  =  945.11
Error =  17.82326 %
Orde  3  Nilai Kecepatan jatuh pada t ke  16  =  1161.93
Error =  18.66065 %
Orde  4  Nilai Kecepatan jatuh pada t ke  16  =  1234.89
Error =  5.90766 %
```

In [49]:

```
tcari = 16
vtcari = 0
for i in range(len(a)):
    vtcari += a[i] * tcari ** i
print("Orde 3. Nilai v(16) adalah : ", vtcari)
```

```
Orde 3. Nilai v(16) adalah :  1234.886601218107
```

TUGAS 5

Linear Regression Using Least Squares

In [50]:

```
import pandas as pd
data = pd.read_csv('/content/headbrain.csv')
data.head()
```

Out[50]:

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590

In [51]:

```
X = data['Head Size(cm^3)'].values
Y = data['Brain Weight(grams)'].values
```

In [52]:

```
def mean(values):
 return sum(values) / float(len(values))

mean_x = mean(X)
mean_y = mean(Y)
n = len(X)
```

Menghitung nilai x2 dan xy

In [53]:

```
num = 0
den = 0
for i in range(n):
  num += (X[i] - mean_x) * (Y[i] - mean_y)
  den += (X[i] - mean_x) ** 2
```

menghitung nilai m

In [54]:

```
m = num / den
print("nilai m : ", m)
```

```
nilai m :  0.26342933948939945
```

menghitung nilai b

In [55]:

```
c = mean_y - (m * mean_x)
print("nilai c : ", c)
```

```
nilai c :  325.57342104944223
```

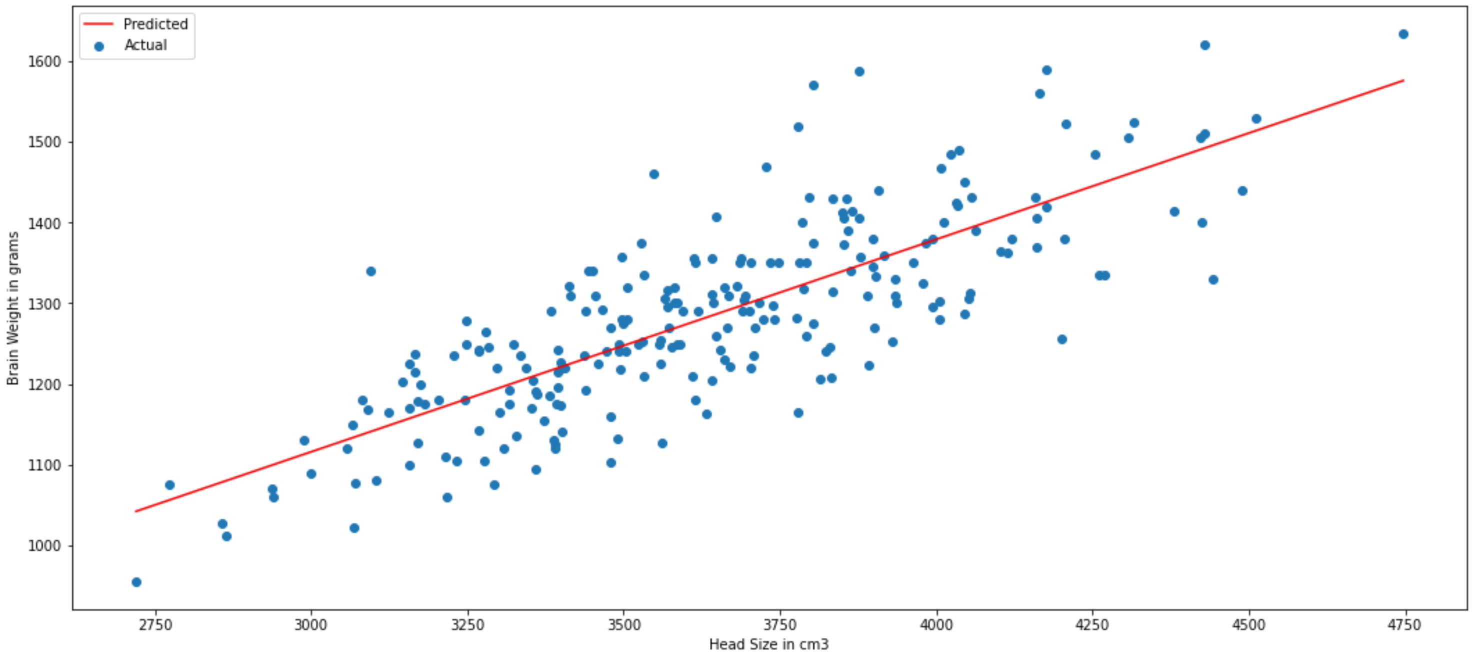
Persamaan Garis

In [56]:

```
Y_pred = m * X + c
```

In [57]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(18,8))
plt.scatter(X, Y, label='Actual')
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red', label='Predicted')
plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```



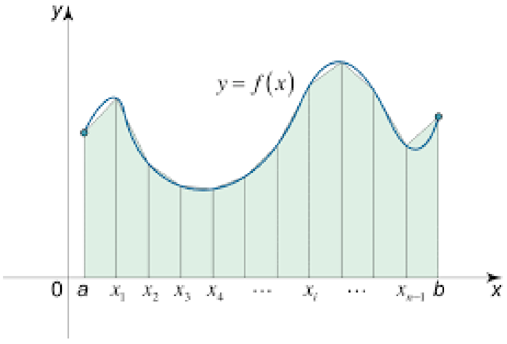
In [58]:

```
import numpy as np
rmse = 0
for i in range(n):
    predictions = list()
    y_pred = c + m * X[i]
    rmse += (Y[i] - y_pred) ** 2
rmse = np.sqrt(rmse/n)
print("RMSE : ", rmse)
```

```
RMSE :  72.1206213783709
```

TUGAS 6

Trapezium Rule



$$L = \frac{h}{2}\left(f_0 + 2\sum_{i=1}^{n-1} f_i + f_n \right)$$

In [59]:

```
def trapesium(x0,xn,n):
    h = (xn - x0) / n
    integral = f(x0) + f(xn)

    for i in range(1,n):
        k = x0 + i*h
        integral = integral + 2 * f(k)

    integral = integral * h/2
    return integral

batas_bawah = float(input("Masukkan batas bawah integral: "))
batas_atas = float(input("Masukkan batas atas integral: "))
interval = int(input("Masukkan jumlah interval: "))
```

```
hasil = trapesium(batas_bawah, batas_atas, interval)
print("Integral hasil dengan aturan trapeium: %0.1f" % (hasil) )
```

```
Masukkan batas bawah integral: 0
Masukkan batas atas integral: 1
Masukkan jumlah interval: 10
Integral hasil dengan aturan trapeium: -1.2
```

Simpson's 1/3 Rule

$$\boxed{\int_a^b f(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]}$$

In [60]:

```
def f(x):
    return 1/(1 + x**2)

def simpson(x0,xn,n):
    h = (xn - x0) / n
    integral= f(x0) + f(xn)

    for i in range(1,n):
        k = x0 + i*h
        if i%2 == 0:
            integral = integral + 2 * f(k)
        else:
            integral = integral + 4 * f(k)

    integral =integral * h/3

    return integral

batas_bawah = float(input("Masukkan batas bawah integral: "))
batas_atas = float(input("Masukkan batas atas integral: "))
interval = int(input("Masukkan jumlah interval: "))

hasil = simpson(batas_bawah, batas_atas, interval)
print("Integral hasil dengan aturan simpson 1/3: %0.6f" % (hasil) )
```

```
Masukkan batas bawah integral: 0
Masukkan batas atas integral: 1
Masukkan jumlah interval: 10
Integral hasil dengan aturan simpson 1/3: 0.785398
```