LAPORAN PRAKTIKUM

METODE NUMERIK

PERSAMAAN NON LINEAR

Disusun Oleh:

Nadya Mumtazah (24060120120027)

Informatika B1

UNIVERSITAS DIPONEGORO

FAKULTAS SAINS DAN MATEMATIKA

PROGRAM STUDI S1 INFORMATIKA

2021

BAB I

PENDAHULUAN

A. Rumusan Masalah

1. Selesaikan sistem persamaan nonlinear berikut :

$$x_1^2 + x_2^2 - 2x_3 = 6$$
$$x_1^2 - 2x_2 + x_3^3 = 3$$
$$2x_1\ x_3 + 3x_2^2 - x_3^2 = -27$$

   Diketahui $x^{(0)} = [1, 1, 1]^T$

2. Selesaikan sistem persamaan nonlinear berikut :

$$3x_1\ - \cos(x_2x_3) - \frac{1}{2} = 0$$
$$x_1^2 - 81(x_2 + 0.1)^2 + \sin\ \ x_3\ + 1.06 = 0$$
$$e^{-x1x2}x_3 + 20x_3\ - \frac{10\pi - 3}{3} = 0$$

   Diketahui $x^{(0)} = [0.1, 0.1, -0.1]^T$

B. Tujuan

Dapat menentukan penyelesaian sistem persamaan nonlinear secara numeric dengan menggunakan metode Newton Raphson dimana didalamnya mengombinasikan metode beda hingga dan metode eliminasi Gauss

C. Dasar Teori

$$f1(x1, \ldots., xn) = 0$$
$$f2(x1, \ldots., xn) = 0$$

$$fn(x1, \ldots., xn) = 0$$

Sebuah sistem n persamaan nonlinear f(x) = 0 dimana x dan f masing masing menunjukkan seluruh vector nilai x_i dan fungsi f_i, i = 0,1,..,n-1 diperoleh secara berulang menggunakan rumus rekursif berikut

$$x^{(k+1)} = x^{(k)} + \delta_x$$

Koreksi δ_x diperoleh dengan memecahkan sistem berikut persamaan aljabar linier

$$J.\delta_x = -f$$

Dimana J adalah matriks Jacobian, dengan $h = \delta_x$. Adapun nilai $J(x)$ dapat dicari dengan metode beda hingga, sehingga tidak perlu mencari turunan analitiknya.

Misal

$$x^2 - y^2 = -4x + 2y + 1$$
$$x^2 = -5y + 4$$

Atau

$$f1(x, y) = x^2 - y^2 + 4x - 2y - 1 = 0$$

$$f2(x,y) = x^2 + 5y - 4 = 0$$

$$J(x,y) = \begin{bmatrix} 2x+4 & -2y-2 \\ 2x & 5 \end{bmatrix}$$

Nilai δ_x dapat diperoleh dengan penyelesaian sistem persamaan linear menggunakan metode eliminasi Gauss atau yang lainnya.

BAB II

PEMBAHASAN

A. Source Code

```
import numpy as np
from scipy.optimize import minimize
```

```
## module gaussElimin
""" x = gaussElimin(a,b).
Menyelesaikan [a]{b} = {x} dengan Eliminasi Gauss  .
"""
import numpy as np
def gaussElimin(a,b):
  n = len(b)
# phase Eliminasi Gauss
  for k in range(0,n-1):
    for i in range(k+1,n):
      if a[i,k] != 0.0:
          lam = a [i,k]/a[k,k]
          a[i,k+1:n] = a[i,k+1:n] - lam*a[k,k+1:n]
          b[i] = b[i] - lam*b[k]
# Substitusi Mundur
  for k in range(n-1,-1,-1):
    b[k] = (b[k] - np.dot(a[k,k+1:n],b[k+1:n]))/a[k,k]
  return b
```

```
## module newtonRaphson2
import numpy as np
import math
def newtonRaphson2(f,x,tol=1.0e-9):
  def jacobian(f,x):
    h = 1.0e-4
    n = len(x)
    jac = np.zeros((n,n))
    f0 = f(x)
    for i in range(n):
```

```
      temp = x[i]
      x[i] = temp + h
      f1 = f(x)
      x[i] = temp
      jac[:,i] = (f1 - f0)/h
    return jac,f0
  for i in range(30):
    jac,f0 = jacobian(f,x)
    if math.sqrt(np.dot(f0,f0)/len(x)) < tol: return x
    dx = gaussElimin(jac,-f0)
    x = x + dx
    if math.sqrt(np.dot(dx,dx)) < tol*max(max(abs(x)),1.0):
      return x
  print("Terlalu banyak iterasi")
```

```
import numpy as np
import math

def f(x):
  f = np.zeros(len(x))
  f[0] = x[0]**2 + x[1]**2 - 2*x[2] - 6
  f[1] = x[0]**2 - 2*x[1] + x[2]**3 - 3
  f[2] = 2*x[0]*x[2] - 3*(x[1]**2) - x[2]**2 + 27
  return f
x0 = np.array([1.0, 1.0, 1.0])
xs = newtonRaphson2(f,x0)
print("Solusi x =",xs)
print("Nilai f pada ",xs, 'adalah ',f(xs))
```

```
import numpy as np
import math

def f(x):
  f = np.zeros(len(x))
  f[0] = 3*x[0] + math.cos(x[1]*x[2]) - 1/2
  f[1] = x[0]**2 - 81*(x[1]+0.1)**2 + math.sin(x[2]) + 1.06
  f[2] = math.exp(-x[0]*x[1]) + 20*x[2] + (math.pi*10-3)/3
  return f
x0 = np.array([0.1, 0.1, -0.1])
xs = newtonRaphson2(f,x0)
print("Solusi x =",xs)
print("Nilai f pada ",xs, 'adalah ',f(xs))
```

B. Penjelasan

Pada percobaan pertama, sistem persamaan non linear yang dimasukkan adalah

$$x_1^2 + x_2^2 - 2x_3 = 6$$
$$x_1^2 - 2x_2 + x_3^3 = 3$$
$$2x_1 \; x_3 + 3x_2^2 - x_3^2 = -27$$

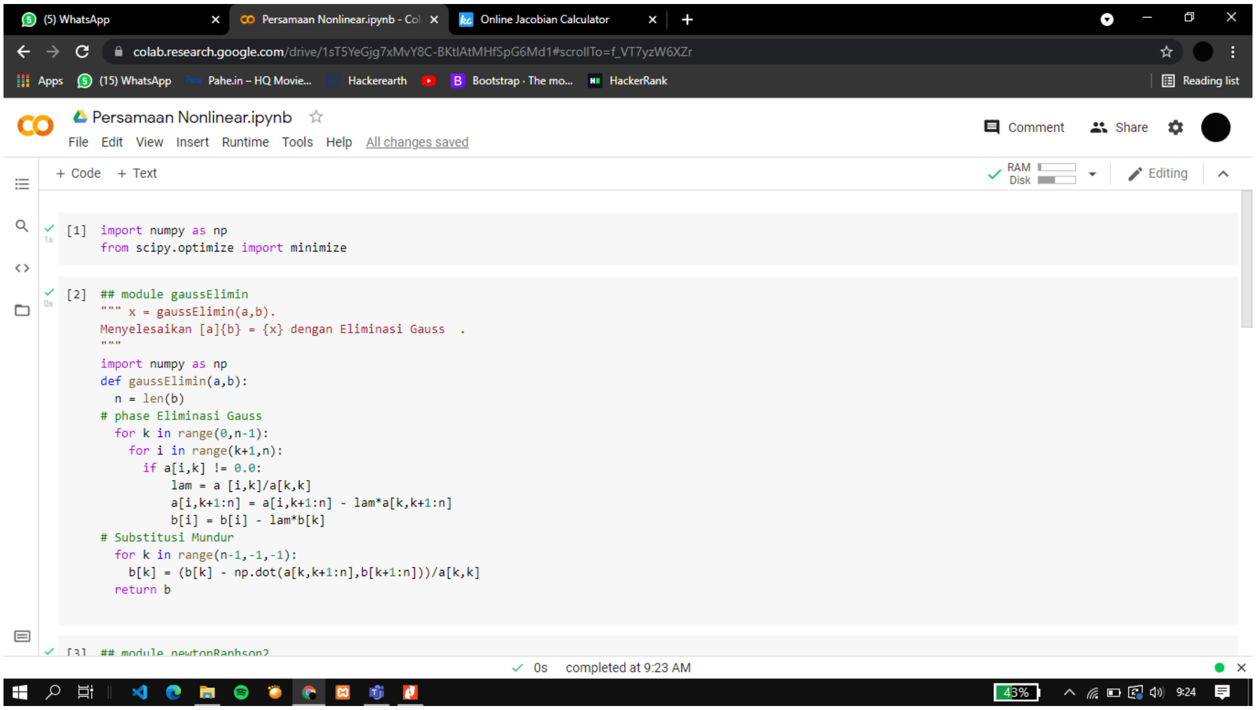
dengan nilai x0 = [1.0, 1.0, 1.0] sehingga menghasilkan solusi x = [1. 3. 2.]. Nilai f pada [1. 3. 2.] adalah `[ 3.24549276e-11  7.06759096e-11 -1.87867499e-11]`.

Pada percobaan kedua, sistem persamaan non linear yang dimasukkan adalah

$$3x_1 \; - \cos(x_2 x_3) - \frac{1}{2} = 0$$
$$x_1^2 - 81(x_2 + 0.1)^2 + \sin \quad x_3 \; + 1.06 = 0$$
$$e^{-x1x2} x_3 + 20x_3 \; - \frac{10\pi - 3}{3} = 0$$

Pada program di atas digunakan math untuk memasukkan nilai pi, sin, cos, dan e pada persamaan, Nilai x0 yang digunakan yaitu [0.1, 0.1, -0.1] sehingga menghasilkan solusi x = [-0.16665665 -0.01480732 -0.52347554]. Nilai f pada `[-0.16665665 -0.01480732 -0.52347554]` adalah `[-1.35047529e-12 -7.98500155e-10 1.36779477e-13]`

C. Screenshot

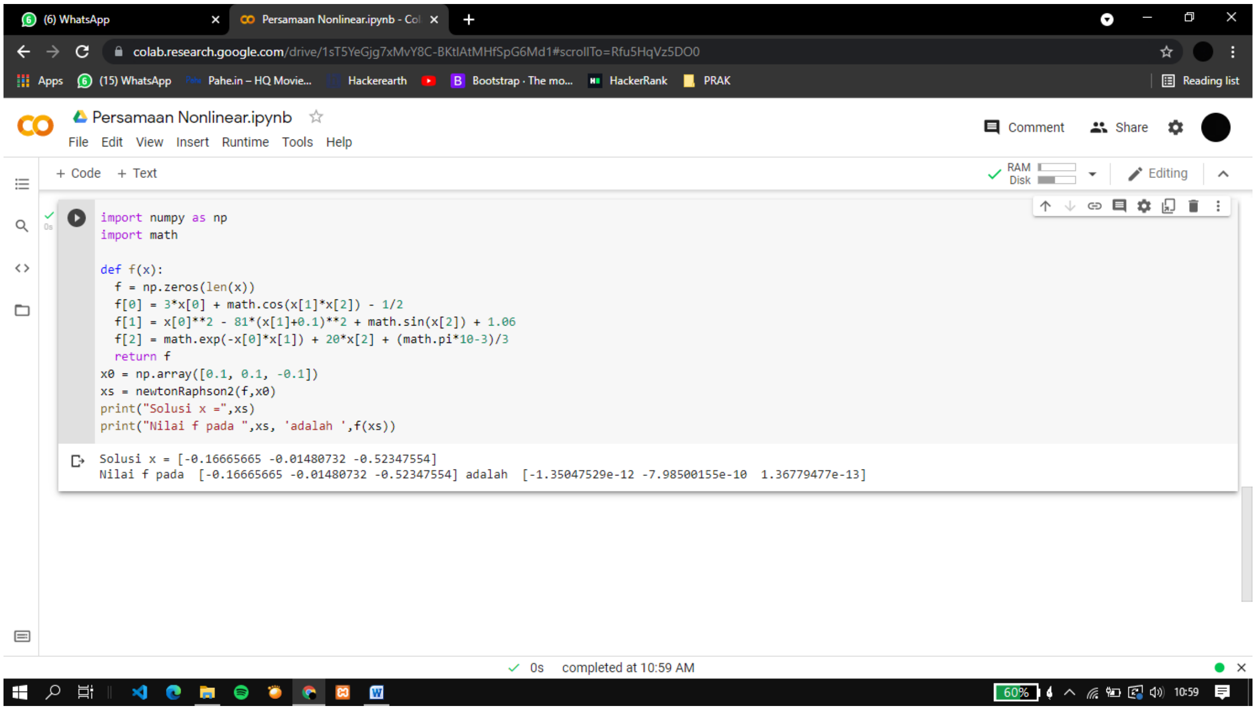


```python
## module newtonRaphson2
import numpy as np
import math
def newtonRaphson2(f,x,tol=1.0e-9):
  def jacobian(f,x):
    h = 1.0e-4
    n = len(x)
    jac = np.zeros((n,n))
    f0 = f(x)
    for i in range(n):
      temp = x[i]
      x[i] = temp + h
      f1 = f(x)
      x[i] = temp
      jac[:,i] = (f1 - f0)/h
    return jac,f0
  for i in range(30):
    jac,f0 = jacobian(f,x)
    if math.sqrt(np.dot(f0,f0)/len(x)) < tol: return x
    dx = gaussElimin(jac,-f0)
    x = x + dx
    if math.sqrt(np.dot(dx,dx)) < tol*max(max(abs(x)),1.0):
      return x
  print("Terlalu banyak iterasi")
```

```python
import numpy as np
import math

def f(x):
  f = np.zeros(len(x))
  f[0] = x[0]**2 + x[1]**2 - 2*x[2] - 6
  f[1] = x[0]**2 - 2*x[1] + x[2]**3 - 3
  f[2] = 2*x[0]*x[2] - 3*(x[1]**2) - x[2]**2 + 27
  return f
x0 = np.array([1.0, 1.0, 1.0])
xs = newtonRaphson2(f,x0)
print("Solusi x =",xs)
print("Nilai f pada ",xs, 'adalah ',f(xs))
```

```
Solusi x = [1. 3. 2.]
Nilai f pada  [1. 3. 2.] adalah  [ 3.24549276e-11  7.06759096e-11 -1.87867499e-11]
```



BAB III

KESIMPULAN

Dalam penyelesaian persamaan non-linier diperlukan akar akar persamaan non-linier dimana akar sebuah persamaan non-linier $f(x) = 0$ merupakan nilai x yang menyebabkan nilai $f(x)$ sama dengan nol. Sistem persamaan non-linier dapat diselesaikan dengan beberapa cara numeric, salah satunya yaitu dengan menggunakan metode Newton Raphson yang didalamnya mengombinasikan metode beda hingga dan eliminasi Gauss.