**BAB I**

**PENDAHULUAN**

1. **Tujuan**

Dapat menghitung akar persamaan nonlinear dengan Metode *Bisection* (Metode Biseksi) dan Metode Newton-Rapson.

1. **Rumusan Permasalahan**
2. Diberikan fungsi terdapat sebuah akar riil dalam selang (-1.0, 1.0). Carilah akar tersebut dengan metode biseksi dengan toleransi kesalahan 1e-5.
3. Diberikan fungsi terdapat sebuah akar riil dalam selang (-1.0, 1.0). Carilah akar tersebut dengan metode newton rapshon dengan toleransi kesalahan 1e-5.

**BAB II**

**DASAR TEORI**

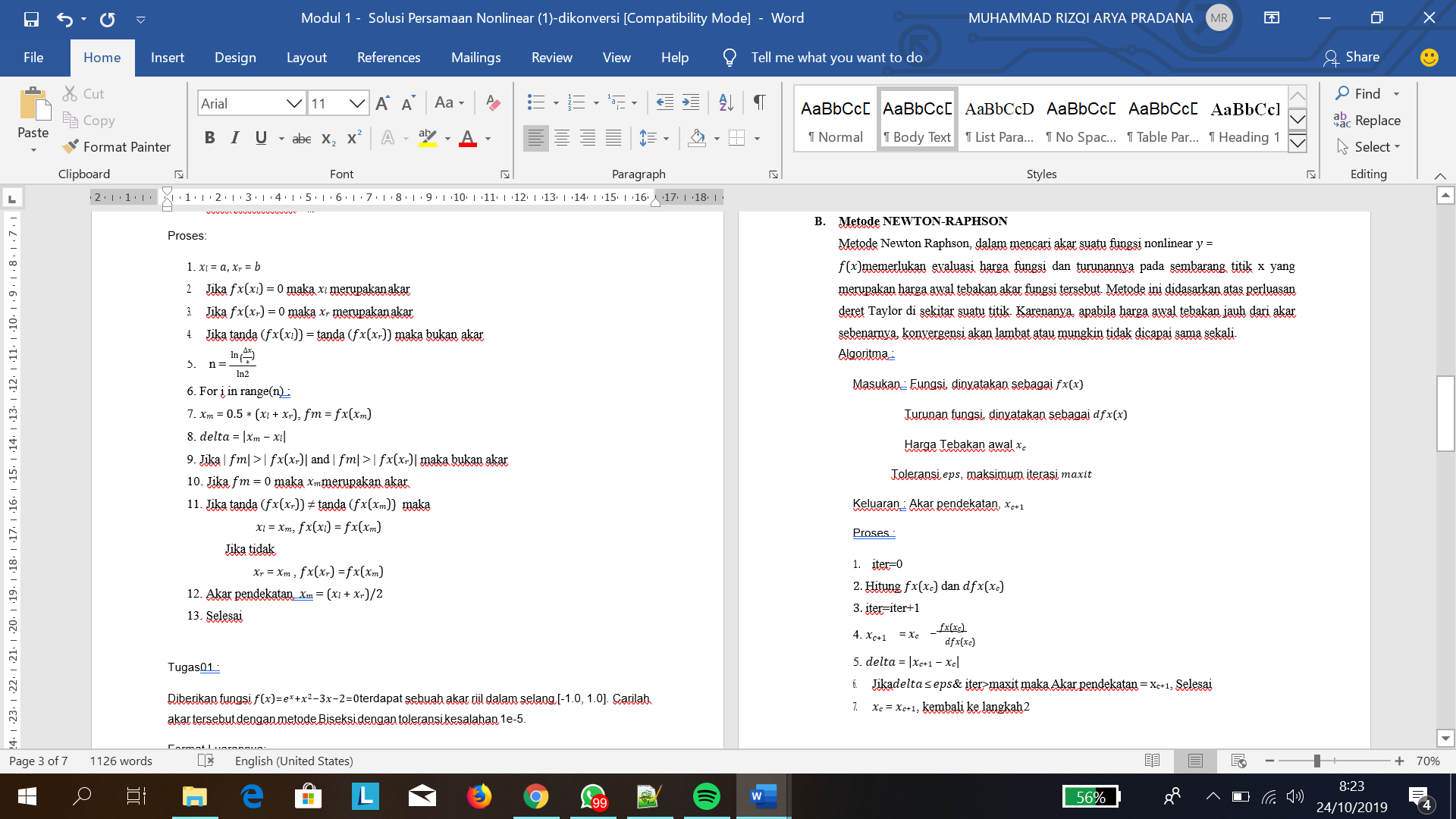
1. **Metode Biseksi**

Dalam metode Biseksi, interval yang mengandung akar dibagi menjadi dua secara berurutan hingga ukuran interval mengecil dan akhirnya mencapai harga toleransi kesalahan yang diinginkan. Dalam interval [a,b] terdapat sebuah akar (yang akan dicari), apabila dipenuhi :

Setiap mengalami lelaran, selang [a,b] dibagi menjadi dua di x = c sehingga terdapat dua buah upaselang yang berukuran sama, yaitu selang [a,c] dan [c,b]. Selang yang diambil untuk lelaran berikutnya adalah upaselang yang memuat akar, bergantung pada apakah atau .

Selang yang baru dibagi menjadi dua lagi dengan cara yang sama. Begitu seterusnya sampai ukuran selang yang baru sudah sangat kecil. Kondisi berhenti lelaran dapat dipilih salah satu dari tiga kriteria berikut:

1. Lebar selang baru : yang dalam hal ini adalah nilai toleransi lebar selang yang mengurung akar.
2. Nilai fungsi di hampiran akar: . Beberapa Bahasa pemrograman membolehkan pembandingan dua buah bilangan riil, sehingga perbandingan dibenarkan. Namun kalua kita kembali ke konsep awal bahwa dua buah bilangan riil tidak dapat dibandingkan kesamaannya karena representasinya di dalam mesin tidak depat, maka kita dapat menggunakan bilangan yang sangat kecil (misalnya epsilon mesin) sebagai pengganti nilai 0. Dengan demikian, menguji kesamaan dapat kita hampiri dengan .
3. Galat relative hampiran akar: adalah galat relative hampiran yang diinginkan

Proses yang terjadi pada metode ini, yaitu:

1. **Metode Newton-Rapshon**

Metode Newton-Raphson, dalam mencari akar suatu fungsi nonlinear memerlukan evaluasi harga fungsi dan turunannya pada sembarang titik x yang merupakan harga awal tebakan akar fungsi tersebut. Metode tersebut berdasar atas perluasan deret Taylor di sekitar suatu titik. Oleh sebab itu, apabila harga awal tebakan jauh dari akar sebenarnya, konvergensi akan lambat atau mungkin tidak dicapai sama sekali.

Proses yang terjadi pada metode ini, yaitu:

1. Iter = 0
2. Hitung fx(xc) dan dfx(xc)
3. Iter = iter + 1
4. Jika maka akar pendekatan . Selesai
5. , kembali ke langkah 2

**BAB III**

**PEMBAHASAN**

1. **Penyelesaian dengan Metode Biseksi**
2. ***Source Code Proses Kerja***

# MUHAMMAD RIZQI ARYA PRADANA , 24060118130119, PRAKTIKUM METODE NUMERIK KELAS C (LAB E DALAM)

# ASPRAK: MBAK ALMA, MBAK GIZKA

# MATERI: BISECT

def bisect(f,a,b,tol=0.00001,maxiter=50):

#\*\*\*

# f: fungsi masukan atau fungsi yang akan dicari akarnya

# a: batas bawah

# b: batas atas

# tol: toleransi

# maxiter: maksimum iterasi

# returnnya: nilai akar persamaan

#\*\*\*

# inisialisasi iterator

# NOTE: kata kunci iter pada modul sudag dipakai sebagai

# build-in function dari Python

iterator=0

# menginisialisasi nilai x\_left (batas kiri)

x\_left=a

x\_right=b

# mencari nilai x\_mid (akar sementara)

x\_mid=(x\_left + x\_right) / 2

# jika f(a)\*f(b)==0, dan f(a)<0 atau f(b) <0, maka akar persamaannya

# adalah nilai a atau b itu sendiri

if f(a)\*f(b)==0:

if f(a)==0:

return a

elif f(b)==0:

return b

# jika nilai f(a)\*f(b)>0 maka ada kesalahan

# ada batas bawah dan atas

elif f(a)\*f(b)>0:

raise ValueError('Nilai f(a) dan f(b) harus mempunyai'

+ 'tanda yang berbeda')

while iterator < maxiter:

# jika nilai ...

# maka akar persamaan ada pada rentang x\_left ... x\_mid

if f(x\_left)\*f(x\_mid)<0:

x\_right=x\_mid

# jika nilai...

# maka akar persamaan ada pada rentang x\_mid ... x\_right

elif f(x\_left)\*f(x\_mid)>0:

x\_left = x\_mid

# jika nilai...

# maka akar persamaan adalah nilai x\_mid itu sendiri

# proses selesai

elif f(x\_left)\*f(x\_mid)==0:

return x\_mid

# jika nilai akar berada pada batas baru

# hitung kembali nilai x\_mid

x\_mid=(x\_left+x\_right)/2

#jangan lupa increment iteratornya

iterator=iterator+1

# menghitung delta mbuh ini apa lali inyong

# mungkin ini error atau apa

delta=abs(x\_right-x\_left)

# jika delta lebih kecil dari nilai toleransi

# maka akar persamaan adalah nilai x\_mid pada iterasi sekarang

# proses selesai

if delta<tol:

return x\_mid

return x\_mid

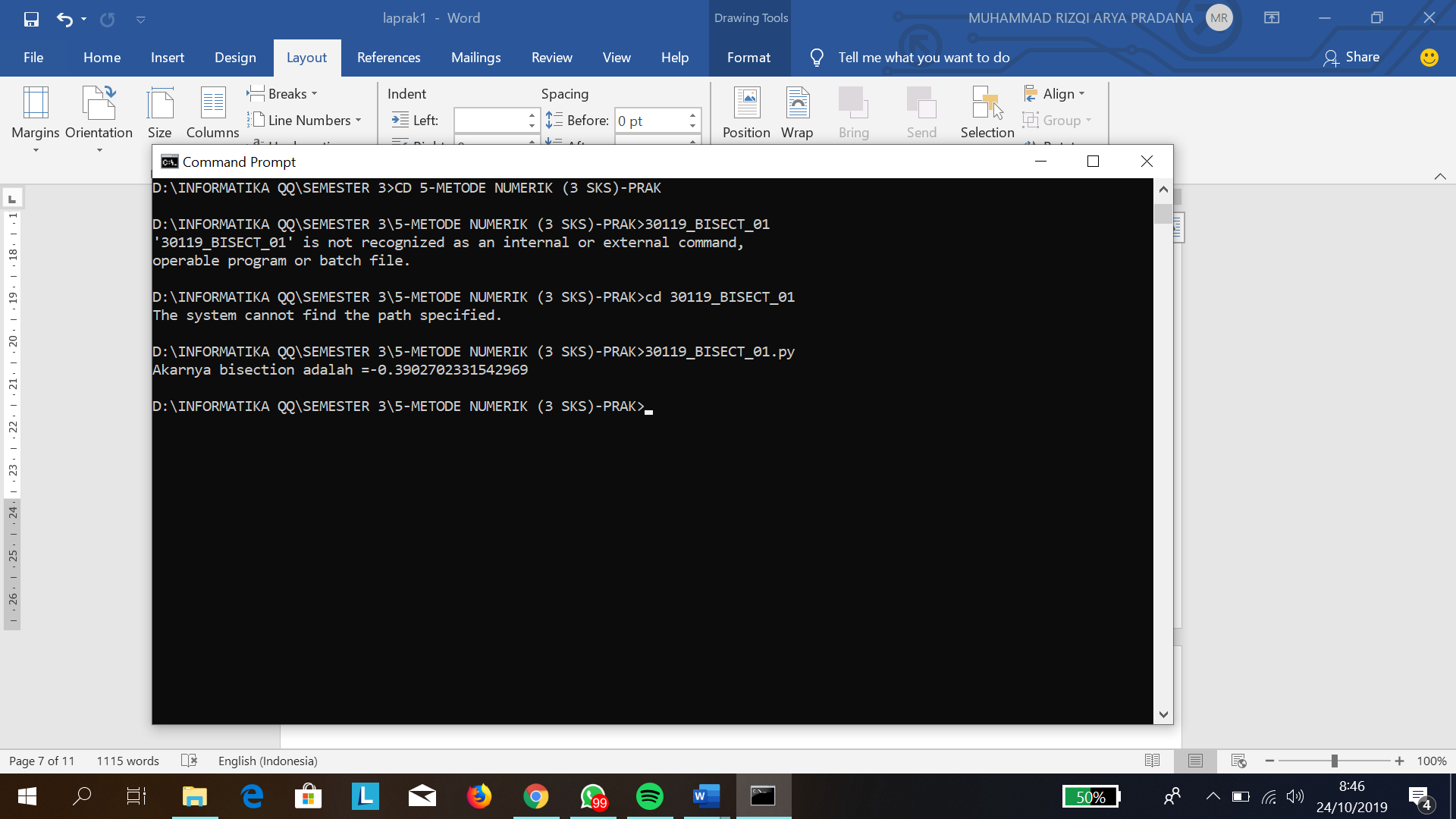
def persamaan(x):

import math

return math.e\*\*x + x\*\*2 - 3\*x - 2

print('Akarnya bisection adalah =' +str(bisect(persamaan,-1,1)))

1. ***Screenshot Hasil Kerja***



1. **Penjelasan Proses dan Hasil Kerja**

Program menampilkan hasil yang diperoleh dari fungsi dengan metode biseksi. Fungsi *bisect* berisi langkah-langkah metode biseksi. Langkah – langkahnya dijelaskan sebagai berikut.

1. Pertama dilakukan pengecekkan apakah batas-batas yang ada mempunyai tanda yang berbeda atau tidak. Apabila tanda berbeda itu berarti memiliki akar.
2. Berikutnya adalah melakukan perulangan yang akan keluar apabila iter (jumlah iterasi yang sudah terjadi) lebih dari maxiter (jumlah maksimal iterasi yang terjadi) atau delta lebih kecil dari toleransi. Fungsi persamaan berisi persamaan yang ingin dicari akar-akarnya. Fungsi ini dapat diganti sesuai dengan persamaan yang ingin dicari akar-akarnya.
3. **Penyelesaian dengan Metode Newton-Rapshon**
4. ***Source Code Proses Kerja***

from math import \*

# MUHAMMAD RIZQI ARYA PRADANA , 24060118130119, PRAKTIKUM METODE NUMERIK KELAS C (LAB E DALAM)

# ASPRAK : MBAK ALMA, MBAK GIZKA

# MATERI : NEWTON-RAPHSON

def newton\_raphson(f,fdef,x0,eps=0.00001,maxiter=100):

"""

f: fungsi masukan atau fungsi yang akan dicari akarnya

fdef: turunan fungsi masukan

x0: tebakan awal

eps: epsilon

maxiter: maksimum iterasi

return: nilai akar persamaan

"""

# inisialisasi iterator

# NOTE: kata kunci iter pada modul sudah dipakai sebagai

# build-in function dari Python

iterator=0

while iterator < maxiter:

x\_f = f(x0)

x\_fdef = fdef(x0)

iterator=iterator+1

x1 = x0- (x\_f/x\_fdef)

delta = abs(x1-x0)

if delta<=eps:

return x1

x0=x1

return x0

def persamaan(x):

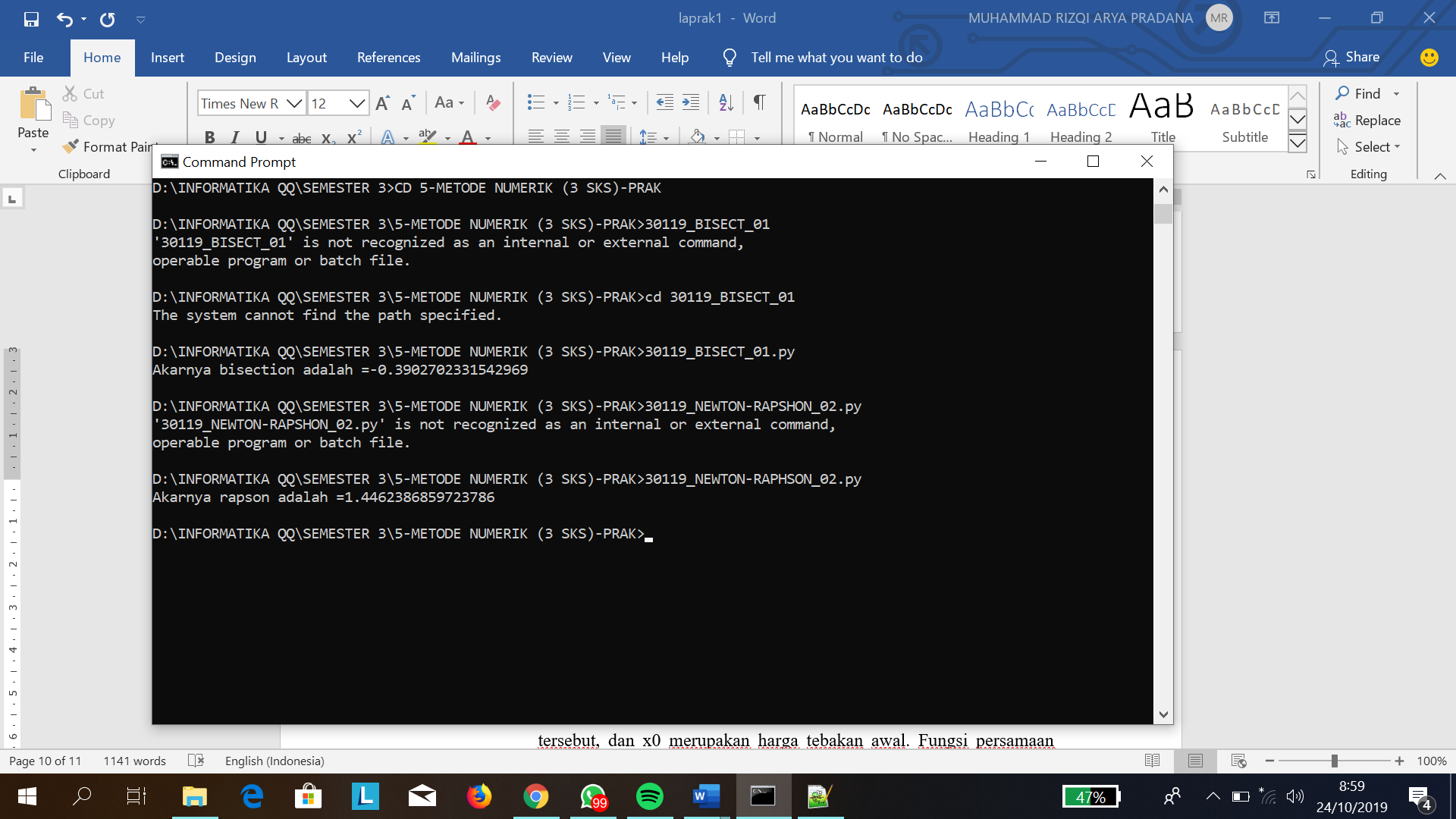
return e\*\*x + x\*\*2 - 3\*x - 2

def turunan(x):

return e\*\*x + 2\*x - 3

print('Akarnya rapson adalah =' +str(newton\_raphson(persamaan,turunan,1)))

1. ***Screenshot Hasil Kerja***



1. **Penjelasan Proses dan Hasil Kerja**

Program menampilkan hasil yang diperoleh dari fungsi dengan metode Newton-Rapshon. Fungsi rapshon berisi langkah-langkah metode newton rapshon.

Masukkan yang terdapat pada fungsi rapshon ada 3, yaitu f, fder, dan x0 dimana f merupakan fungsi persamaan yang dicari, fder merupakan fungsi turunan persamaan tersebut, dan x0 merupakan harga tebakan awal.

Fungsi persamaan berisi persamaan yang akan dimasukkan pada fungsi rapshon dan fungsi turunan berisi turunan dari persamaan yang terdapat pada fungsi persamaan.

Fungsi persamaan dan turunan dapat diganti sesuai dengan persamaan yang ingin dicari akar-akarnya menggunakan metode Newton - Raphson.

**BAB IV**

**PENUTUP**

1. **Kesimpulan**

Akar yang terdapat dari suatu persamaan nonlinear dapat dicari dengan Metode *Bisection* (Biseksi) dan Metode Newton-Raphson. Langkah-langkah yang tersedia dapat dibuat dengan menggunakan Bahasa *Python* sehingga tidak perlu melakukan iterasi secara sistem manual seperti yang ada pada *source code* di atas. Hasil kerja dari sistem program tersebut adalah sebagai berikut.

1. Pada soal pertama tentang implementasi Metode *Bisection*, dicari akar dari persamaan dengan batas kiri -1 dan batas kanan 1. Hasil yang diperoleh menggunakan metode biseksi yang telah dibuat dalam Bahasa *Python* adalah -0.3902702331542969.
2. Pada soal kedua tentang implementasi Metode Newton-Raphson, dicari akar dari persamaan yang memiliki turunan dan harga taksiran 1. Hasil yang diperoleh menggunakan Metode Newton-Raphson yang telah dibuat dalam Bahasa *Python* adalah 1.4462386859723786.

**DAFTAR PUSTAKA**

Munir, Renaldi. 2015. *Metode Numerik*. Bandung: Informatika

Universitas Diponegoro. 2019. *Modul I Solusi Persamaan Nonlinear.* Semarang: Universitas Diponegoro