

Lembar Screenshot Hasil Running

Dwi Cahyo Rikie Hady Cumoro
NIM 1709075050

Program Studi Teknik Elektro

Menggunakan Metode Bisection, Gauss Seidel, dan Newton Raphson
Metode Biseksi

The screenshot shows the OnlineGDB interface with a Python program implementing the Bisection method. The code defines a function `bisection(f, a, b, e=0.5, N=100)` that iteratively narrows down the root of a function `f` between `a` and `b` until the absolute value of the function is less than `e` or the number of iterations reaches `N`. The function `f` is defined as `2*x**3 - 4*x**2 - 24` with `a = 2`. The program runs successfully, outputting the result `3.1758` after `8` iterations.

```
main.py
1 def bisection(f, a, b, e=0.5, N=100):
2     x_a = a
3     x_b = b
4
5     if f(x_a)*f(x_b) >= 0:
6         print("Use another guess range")
7         return None
8
9     for n in range(1, N):
10        x_m = (x_a + x_b)/2
11        if abs(f(x_m)) < e:
12            return x_m, n
13        else:
14            if f(x_a)*f(x_m) < 0:
15                x_b = x_m
16            elif f(x_b)*f(x_m) < 0:
17                x_a = x_m
18        return x_m, n
19
20
21 f = lambda x: 2*x**3 - 4*x**2 - 24
22 a = 2

input
Result : 3.1758
In 8 iteration
...Program finished with exit code 0
Press ENTER to exit console.
```

Metode Gauss Seidel

The screenshot shows the OnlineGDB interface with a Python program implementing the Gauss-Seidel method for solving a system of linear equations. The system is represented by matrix `A` and vector `b`. The program iterates until the error is less than the tolerance (0.5). The output shows the iteration process and the final solution values for `x1`, `x2`, and `x3`.

```
main.py
6 # [-1*x1 + 5*x2 - 8*x3 = 5]
7
8 BATAS_ITERASI = 50 #Batas Iterasi
9 print("Menampilkan Matriks :")
10 A = np.array([[3., 1., -1.],
11               [2., 4., -1.],
12               [-1., 5., -8.], 1])
13 b = np.array([1.0, 5.0, 5.0])
14
15 # Mencari Nilai X1, X2, X3, dengan menggunakan metode Gauss Seidel
16 for i in range(A.shape[0]):
17     row = ["{0:3g}*x{1} ".format(A[i, j], j + 1) for j in range(A.shape[1])]
18     print("[{0}] = [{1:3g}] ".format(" ".join(row), b[i]))
19
20 x = np.zeros_like(b)
21
input
Menampilkan Matriks :
[ 3*x1 + 1*x2 + -1*x3] = [ 1]
[ 2*x1 + 4*x2 + -1*x3] = [ 5]
[-1*x1 + 5*x2 + -8*x3] = [ 5]
Iterasi 1: [0. 0. 0.]
Iterasi 2: [0.33333333 1.08333333 0.01041667]
Iterasi 3: [-0.02430556 1.26475694 0.16851128]
Batas Toleransi 0.5
Hasil(x1, x2, x3) : [-0.02430556 1.26475694 0.16851128]
Nilai Error: [ 0.02332899 -0.15809462  0.        ]
...Program finished with exit code 0
Press ENTER to exit console.
```

Lembar Screenshot Hasil Running

Dwi Cahyo Rikie Hady Cumoro

NIM 1709075050

Program Studi Teknik Elektro

Menggunakan Metode Bisection, Gauss Seidel, dan Newton Raphson

Newton Raphson

The screenshot displays the OnlineGDB beta web interface. The browser tabs include 'UTS-METNUM/Newton Raphson', '(17) WhatsApp', 'MOLS', and 'GDB online Debugger | Compiler'. The address bar shows 'onlinegdb.com'. The interface features a sidebar with navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main area shows a Python script for the Newton-Raphson method. The script defines a function `newtons(f, df, x0)` that iteratively finds the root of a function `f` using its derivative `df`. The function `f` is defined as `2*x**3 - 4*x**2 - 24` and `df` as `6*x**2 - 8*x`. The initial guess `x0` is 5. The output shows the iterative process converging to the root 3.1838 after 4 iterations.

```
1 def newtons(f, df, x0):
2     e = 0.001
3     N = 100
4     for i in range(0, N):
5         print("%d | %f | %f" % (i, x0, f(x0)))
6         if abs(f(x0)) < e:
7             return x0, i
8         if df(x0) == 0:
9             print("Zero derivative")
10            return None
11            x0 = x0 - f(x0)/df(x0)
12            print("Maximum iteration")
13            return x0, i
14
15 f = lambda x: 2*x**3 - 4*x**2 - 24
16 df = lambda x: 6*x**2 - 8*x
17 x0 = 5
18
```

input

```
0 | 5.000000 | 126.000000
1 | 3.854545 | 31.107895
2 | 3.321043 | 5.140410
3 | 3.191259 | 0.263886
4 | 3.183841 | 0.000833
Result : 3.1838
In 4 iteration
...Program finished with exit code 0
Press ENTER to exit console.
```

Windows taskbar at the bottom shows the date and time as 8:05 AM 10/15/2021, and the weather as 33°C Hujan.