HW1

October 2, 2024

1 HW 1 - root finding methods

Anna Dodson September 30, 2024

A1. Newton-Raphson Method

[ ]:
```
initial_guess = -1.6
import numpy as np

def f(x):
    return x * np.sin(3 * x) - np.exp(x)

def df_dx(x):
    return np.sin(3 * x) + 3 * x * np.cos(3 * x) - np.exp(x)

tolerance = 1e-6
max_iter = 100
initial_guess = -1.6

def iter_newton_raphson(f, df_dx, x0, tol, max_iter) -> (float, int):
    xn = x0
    for n in range(max_iter):
        fxn = f(xn)
        print(f"Iteration {n + 1}: x_n is {xn}, f(x) is {fxn}")
        if abs(fxn) < tol:
            return xn, n + 1
        dfxn = df_dx(xn)
        if dfxn == 0:
            return None, n + 1  # Derivative is zero (stagnation)
        xn = xn - fxn / dfxn
    return None, max_iter

solution_nr, iterations_nr = iter_newton_raphson(f, df_dx, initial_guess,␣
 ↪tolerance, max_iter)
print(f"Newton-Raphson method: x_r = {solution_nr}, iterations =␣
 ↪{iterations_nr}")
```

```
Iteration 1: x_n is -1.6, f(x) is -1.7957598921320004
Iteration 2: x_n is 3.1979951385210694, f(x) is -25.021941283956743
```

```
Iteration 3: x_n is 2.4644024441424284, f(x) is -9.549068748622544
Iteration 4: x_n is 1.2035359007112925, f(x) is -3.875885058382139
Iteration 5: x_n is 0.6502014632644292, f(x) is -1.312061395158699
Iteration 6: x_n is -0.1169233418248703, f(x) is -0.8494760686598173
Iteration 7: x_n is -0.6605234854521386, f(x) is 0.08899499862097715
Iteration 8: x_n is -0.5219265439062168, f(x) is -0.07145630442850248
Iteration 9: x_n is -0.5665527428708069, f(x) is -0.005622980486580498
Iteration 10: x_n is -0.5707465821813341, f(x) is -5.653766944091476e-05
Iteration 11: x_n is -0.57078961788788, f(x) is -6.016180886803113e-09
Newton-Raphson method: x_r = -0.57078961788788, iterations = 11
```

A2. Bisection Method

[ ]:
```python
# Bisection method parameters
x_left = -0.7
x_right = -0.4

# Bisection method
def bisection(f, x_left, x_right, tol, max_iter):
    midpoints = []
    for n in range(max_iter):
        x_mid = (x_left + x_right) / 2
        f_mid = f(x_mid)
        midpoints.append(x_mid)
        print(f"Iteration {n + 1}: x_mid is {x_mid}, f(x_mid) is {f_mid}")
        if abs(f_mid) < tol:
            return x_mid, n + 1, midpoints  # Root found

        # Check the sign of the function at the midpoint
        if f(x_left) * f_mid < 0:
            x_right = x_mid
        else:
            x_left = x_mid

    return None, max_iter, midpoints

# Run the Bisection method
solution_bisection, iterations_bisection, midpoints_bisection = bisection(f,␣
 ↪x_left, x_right, tolerance, max_iter)
print(f"Bisection method: Solution = {solution_bisection}, Iterations =␣
 ↪{iterations_bisection}")
```

```
Iteration 1: x_mid is -0.55, f(x_mid) is -0.02867404473083124
Iteration 2: x_mid is -0.625, f(x_mid) is 0.06104218498706837
Iteration 3: x_mid is -0.5875, f(x_mid) is 0.021022783912400866
Iteration 4: x_mid is -0.5687500000000001, f(x_mid) is -0.0026924412574776957
Iteration 5: x_mid is -0.578125, f(x_mid) is 0.009458341688780236
Iteration 6: x_mid is -0.5734375, f(x_mid) is 0.0034550269976505454
Iteration 7: x_mid is -0.5710937500000001, f(x_mid) is 0.0003991584435431017
```

```
Iteration 8: x_mid is -0.5699218750000001, f(x_mid) is -0.0011421943170324411
Iteration 9: x_mid is -0.5705078125, f(x_mid) is -0.0003704037468809096
Iteration 10: x_mid is -0.57080078125, f(x_mid) is 1.4656197581341956e-05
Iteration 11: x_mid is -0.570654296875, f(x_mid) is -0.0001778041000428665
Iteration 12: x_mid is -0.5707275390625, f(x_mid) is -8.155652786479006e-05
Iteration 13: x_mid is -0.57076416015625, f(x_mid) is -3.3445808711007885e-05
Iteration 14: x_mid is -0.570782470703125, f(x_mid) is -9.393716383421236e-06
Iteration 15: x_mid is -0.5707916259765625, f(x_mid) is 2.6315129033616103e-06
Iteration 16: x_mid is -0.5707870483398438, f(x_mid) is -3.381033664928701e-06
Iteration 17: x_mid is -0.5707893371582031, f(x_mid) is -3.747433618972451e-07
Bisection method: Solution = -0.5707893371582031, Iterations = 17
```