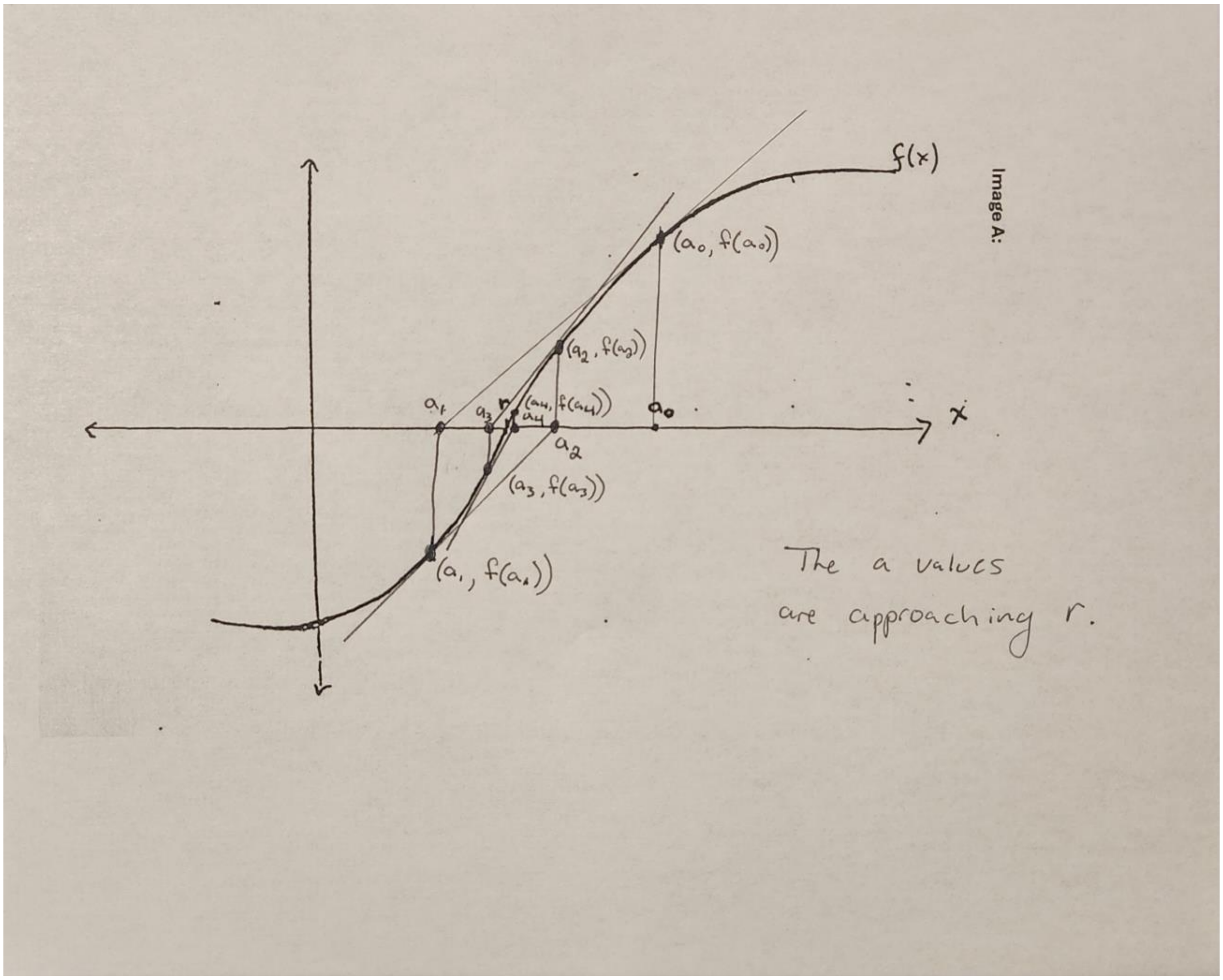
Image A:



f(x)
(a₀, f(a₀))
(a₂, f(a₂))
(a₄, f(a₄))
a₁
a₃
r
a₄
a₂
a₀
x
(a₃, f(a₃))
(a₁, f(a₁))
The a values
are approaching r.

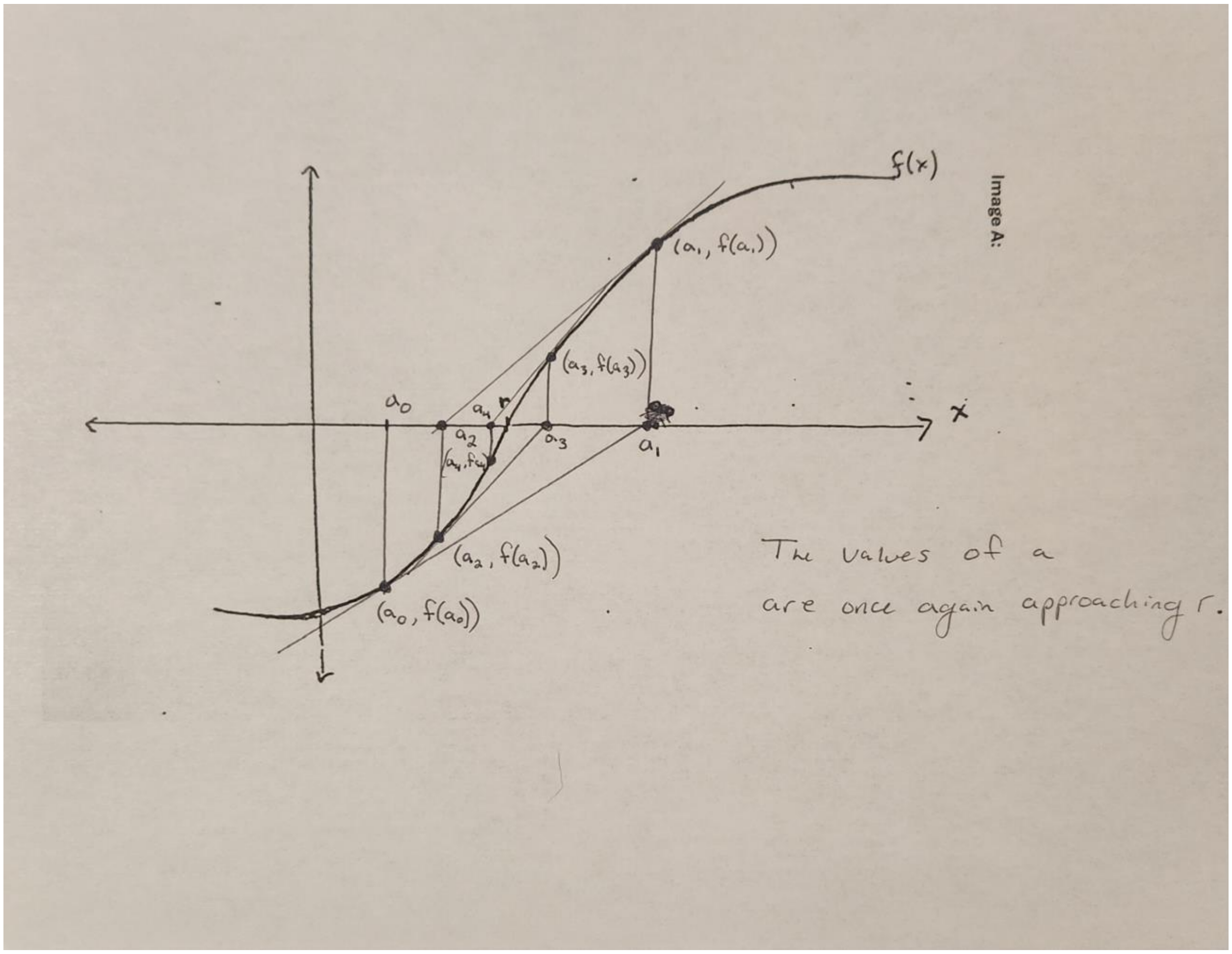
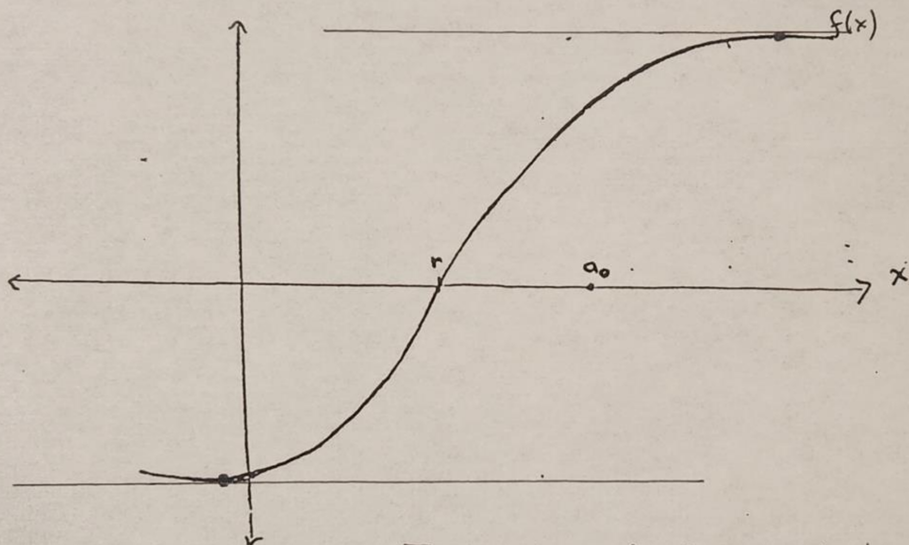


Image A:
$f(x)$
$(a_1, f(a_1))$
$(a_3, f(a_3))$
a_0
a_4
r
a_2
a_3
a_1
x
$(a_4, f(a_4))$
$(a_2, f(a_2))$
$(a_0, f(a_0))$
The values of a
are once again approaching r.

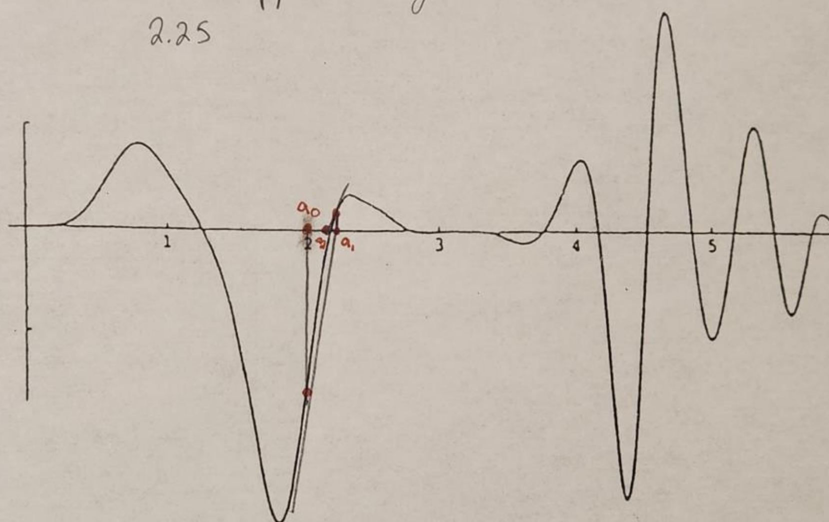
Image A:

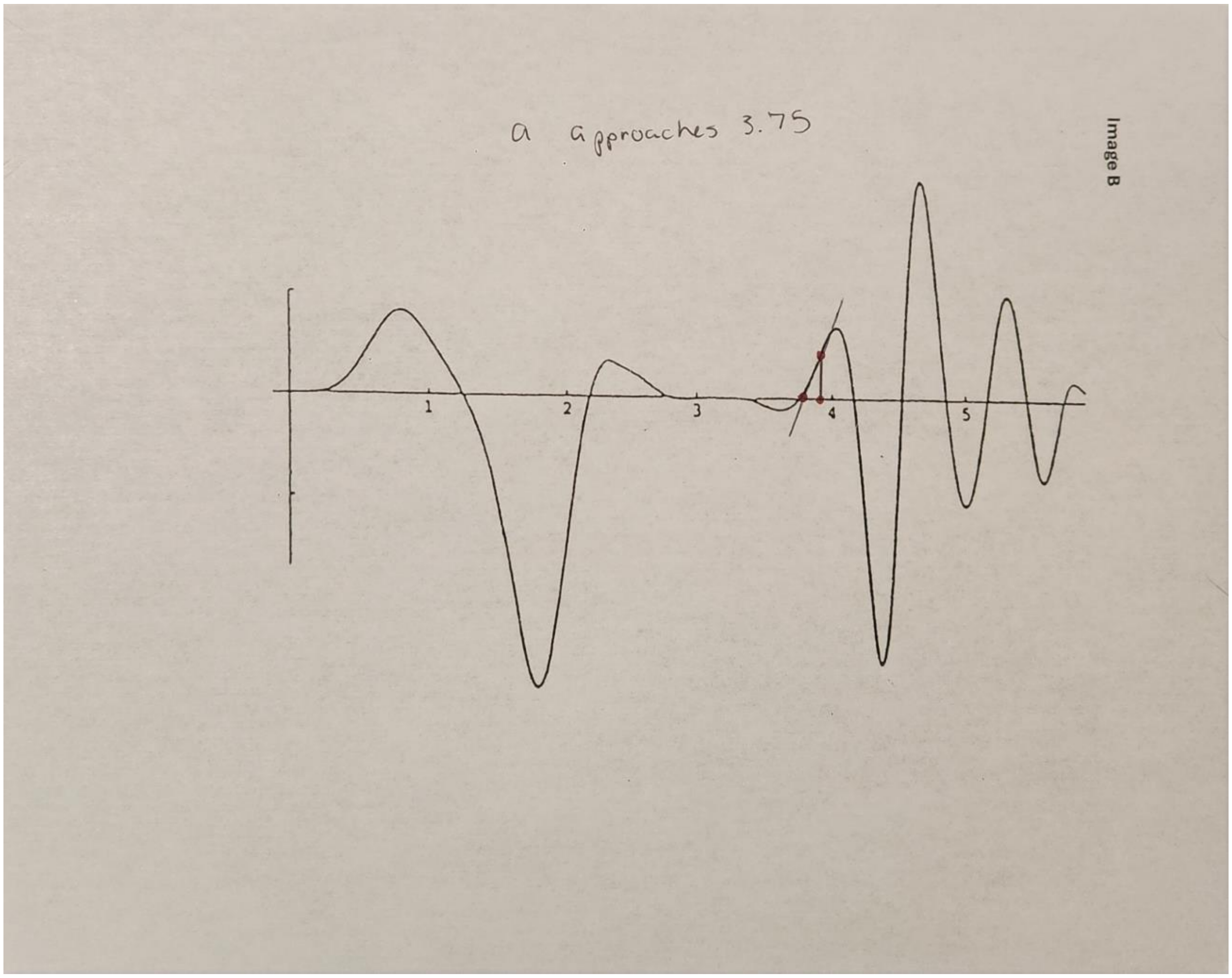


The Starting point can not be where the derivative is 0 because the tangent line will never cross the x axis. The error returned should tell the user to pick a Starting point where the derivative does not equal 0. The logical next step is for the user to pick a different point close to their original selection that has a non-zero slope.

Image B

a is approaching the root 2.25

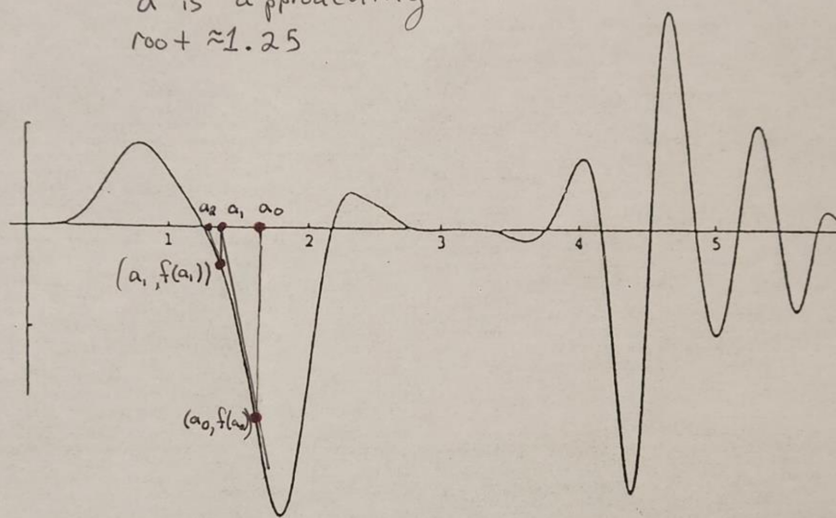




a approaches 3.75
1
2
3
4
5
Image B

Image B

a is approaching the root ≈1.25



NewtonsMethod

October 31, 2024

[30]:
```python
def f(x):
    return (x**2) / 4 + x / 4 - 5

def df(x):
    return 0.5 * x + 0.25

def newtons_method(starting_point, tolerance = 1e-6, max_iterations = 100):
    x = starting_point
    # Error catch for a 0 derivative value
    if x == -0.5:
        return f"Derivative too close to zero. Select another value."
    for i in range(max_iterations):
        derivative = df(x)

        # Newton's method formula
        next_x = x - f(x) / derivative

        if abs(next_x - x) < tolerance:
            return f"Root found at x = {next_x} after {i + 1} iterations."
        x = next_x
    return "Max iterations reached without convergence."

# User input for starting point
user_starting_point = float(input("Enter a starting point for Newton's method: 
↪"))
user_result = newtons_method(user_starting_point)
print(f"Result with starting point {user_starting_point}: {user_result}")
```

```
Enter a starting point for Newton's method:  5
```

```
Result with starting point 5.0: Root found at x = 4.000000000000001 after 4
iterations.
```

[23]:
```python
# Testing with random inputs to find the 2 roots and the number of iterations
test_points = [-10, -4, 0, 3, 6]
for point in test_points:
    result = newtons_method(point)
    print(f"Starting point {point}: {result}")
```

```
Starting point -10: Root found at x = -5.000000000000044 after 5 iterations.
Starting point -4: Root found at x = -5.000000000000032 after 4 iterations.
Starting point 0: Root found at x = 4.0 after 8 iterations.
Starting point 3: Root found at x = 4.000000000000032 after 4 iterations.
Starting point 6: Root found at x = 4.0 after 5 iterations.
```

[33]:

```
# The point that will cause an error is -0.5. If a user enters those an error␣
↪message occurs
user_error_point = float(input("Enter a starting point for Newton's method: "))
user_error_result = newtons_method(user_error_point)
print(f"Result with starting point {user_error_point}: {user_error_result}")
```

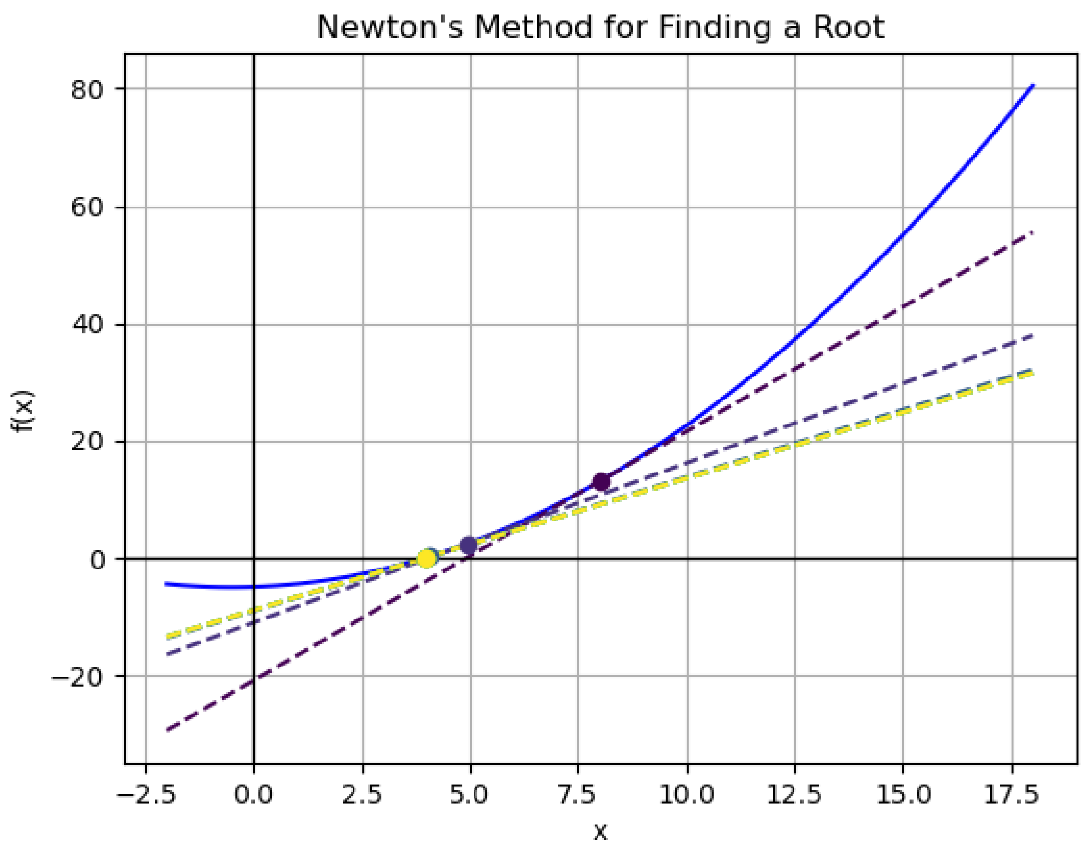
```
Enter a starting point for Newton's method:  -0.5
```

```
Result with starting point -0.5: Derivative too close to zero. Select another
value.
```

[28]:

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm  # For color maps

def Newton_Graph(start):

    steps = 8
    x = start
    window = 10

    x_vals = np.linspace(x - window, x + window, 100)
    y_vals = f(x_vals)

    cmap = cm.get_cmap('viridis', steps)

    plt.plot(x_vals, y_vals, label="Function", color='blue')

    for n in range(steps):
        f_x = f(x)
        f_prime_x = df(x)

        tangent_y = f_x + (f_prime_x * (x_vals - x))

        plt.plot(x_vals, tangent_y, color=cmap(n), label=f"Tangent at x={x:.
↪3f}", linestyle='--')

        plt.scatter(x, f_x, color=cmap(n), label=f"Point ({x:.3f}, {f_x:.3f})",␣
↪zorder=5)

        x = x - f_x / f_prime_x

    plt.axhline(0, color='black', lw=1)
```

```
    plt.axvline(0, color='black', lw=1)
    plt.title("Newton's Method for Finding a Root")
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.grid(True)

    plt.show()

Newton_Graph(8)
```



[ ]: