Problem 1

In [22]:
```
import math

def Numerical_Point_Derivative():
    func_str = input("Enter the function in terms of x: ")
    func = eval(f"lambda x: {func_str}")
    a = float(input("Enter the x value at which you want to find the derivative: ")
    h = 1e-5

    #Using the central difference formula
    derivative = (func(a + h) - func(a - h)) / (2 * h)
    return derivative

print("The derivative of the function at the given point is:", Numerical_Point_Deri
```

```
The derivative of the function at the given point is: -0.2500000000099645
```

In [23]:
```
print("The derivative of the function at the given point is:", Numerical_Point_Deri
```

```
The derivative of the function at the given point is: -3.8341048184897804
```

In [25]:
```
print("The derivative of the function at the given point is:", Numerical_Point_Deri
```

```
The derivative of the function at the given point is: 120000.00006519257
```

In [26]:
```
print("The derivative of the function at the given point is:", Numerical_Point_Deri
```

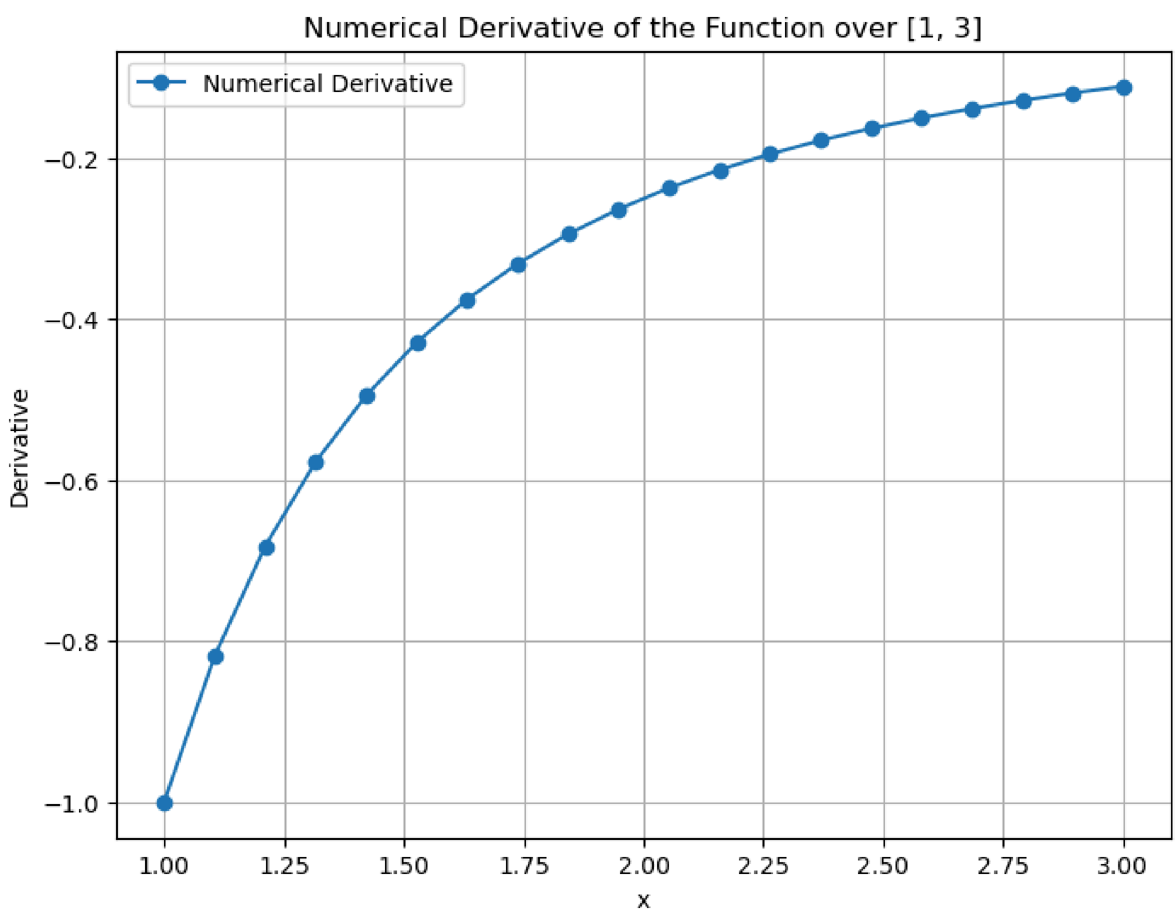
```
The derivative of the function at the given point is: 22.180709777153137
```

Problem 2 The online calculator that I found was the WolframAlpha Numercial Derivative Calculator (https://www.wolframalpha.com/widgets/view.jsp?id=a278064e754d61cbecc14f913b8d5295). When comparing my results to the online calculators results I found that I recieved similar results. One thing I did notice is that the online calculator seemed to avoid using decimals and kept the answers whole numbers.

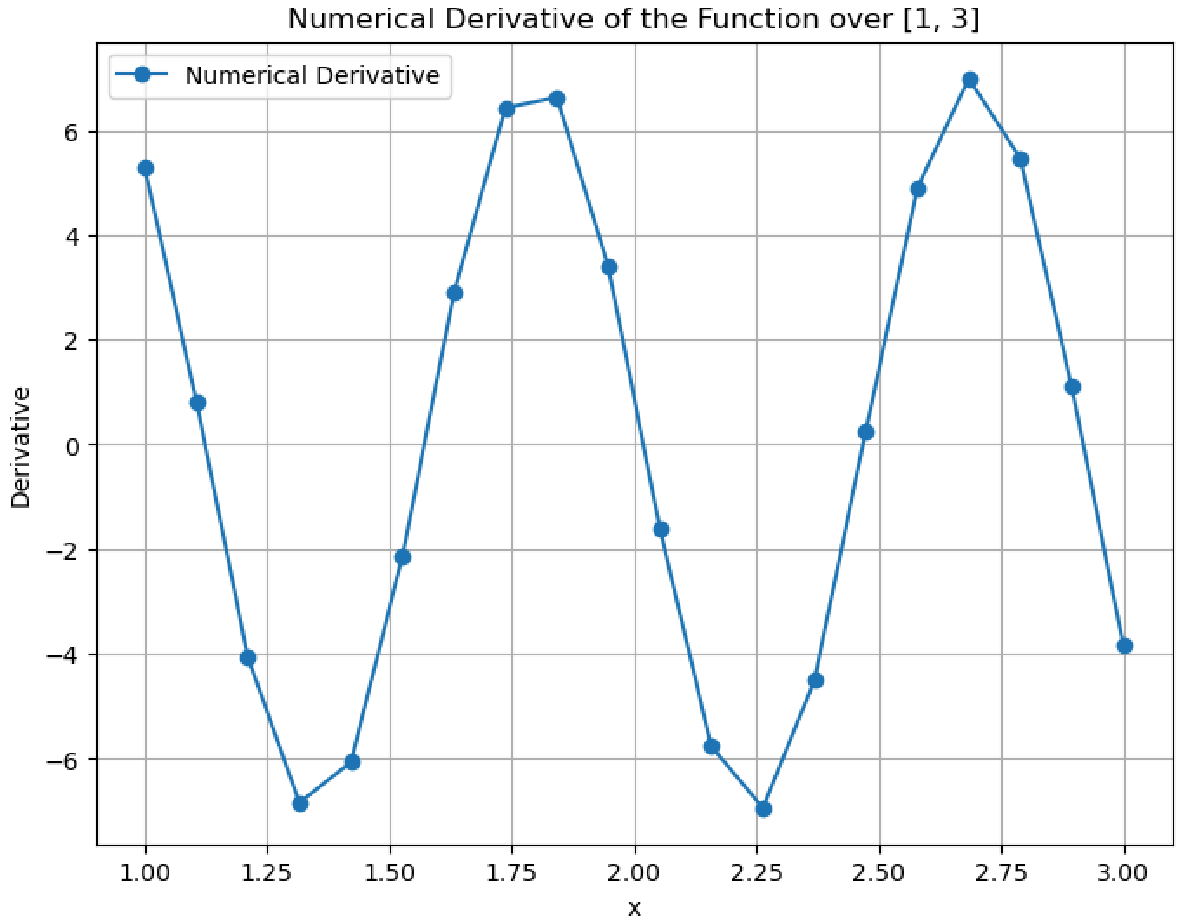
Problem 4

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
def Numerical_Function_Derivative():
    h = 1e-5
    func = input("Enter an algebraic function in terms of x ('x**2', 'math.sin(x)',

    #Convert the input into a lambda function
    f = lambda x: eval(func)

    #generate 20 points from 1 to 3
    Xvals = np.linspace(1, 3, 20)
    DeriVals = []
    #Calculate the derivative
```

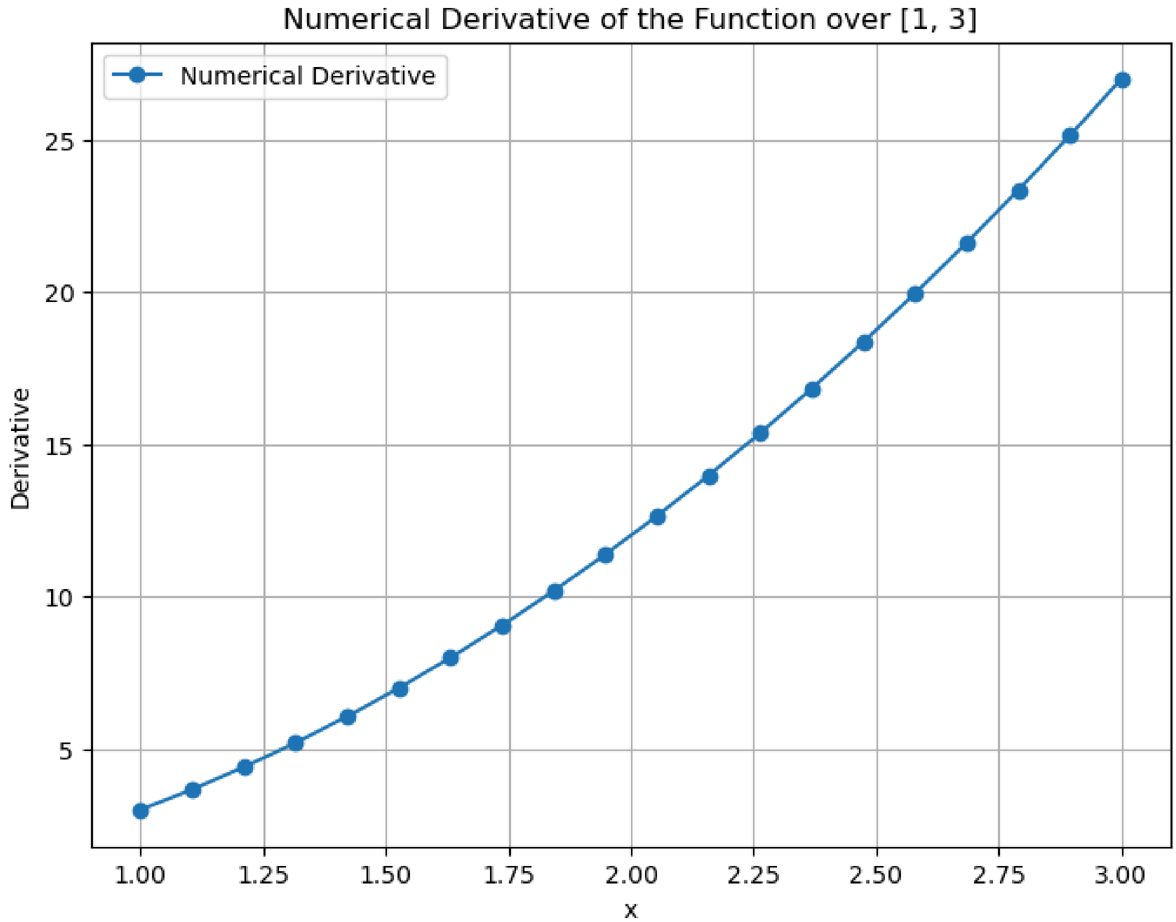
```
    for a in Xvals:
        q = (f(a + h) - f(a)) / h
        DeriVals.append(q)

    #plot results using matplotlib
    plt.figure(figsize=(8, 6))
    plt.plot(Xvals, DeriVals, marker='o', label='Numerical Derivative')
    plt.title("Numerical Derivative of the Function over [1, 3]")
    plt.xlabel("x")
    plt.ylabel("Derivative")
    plt.grid(True)
    plt.legend()
    plt.show()

Numerical_Function_Derivative()
```



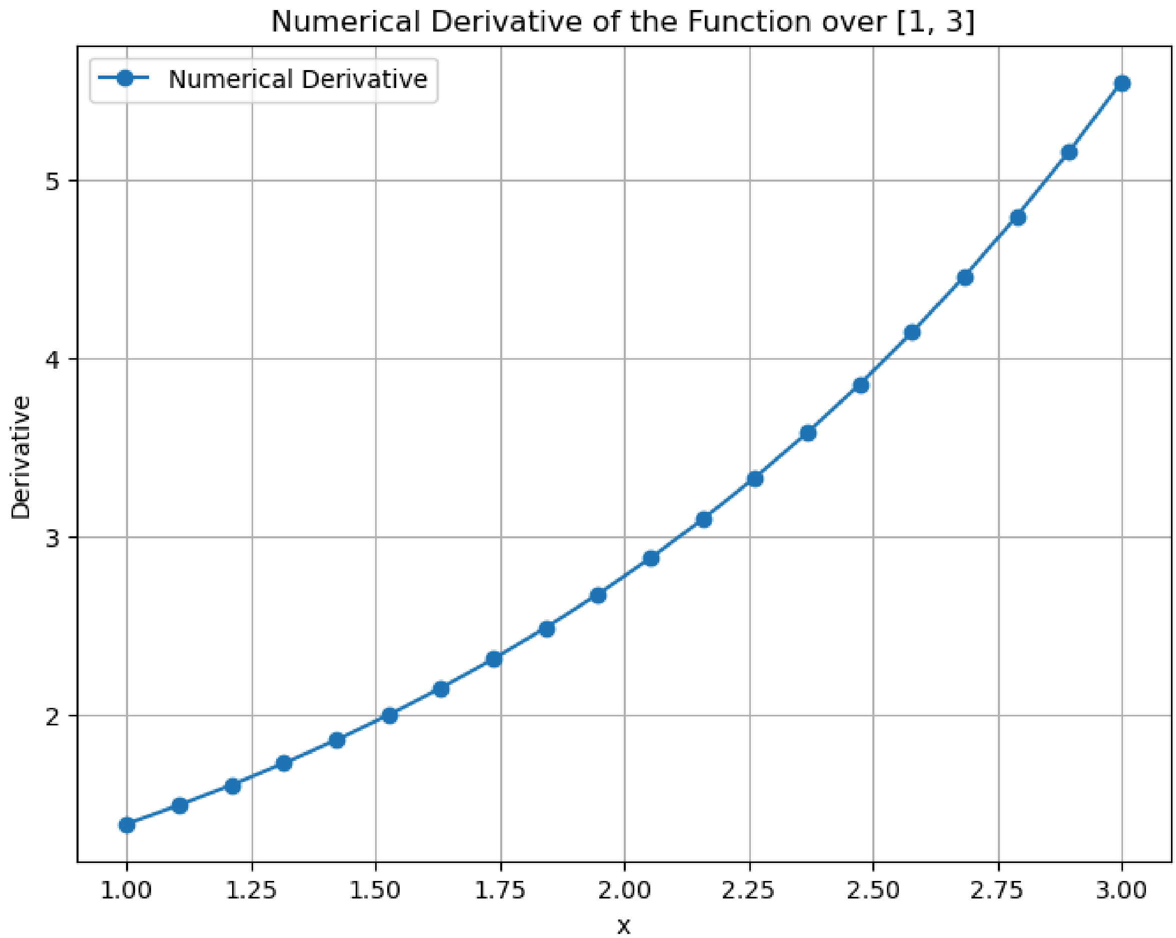
In [14]:
```
Numerical_Function_Derivative()
```



In [17]:
```
Numerical_Function_Derivative()
```



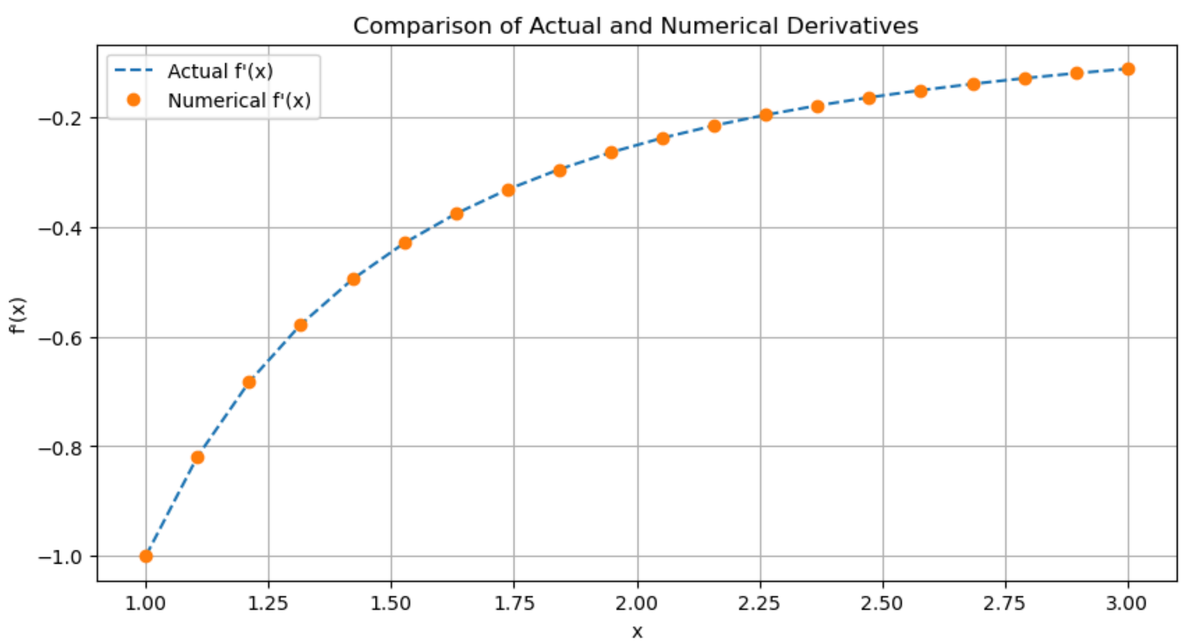
In [18]:
```
Numerical_Function_Derivative()
```

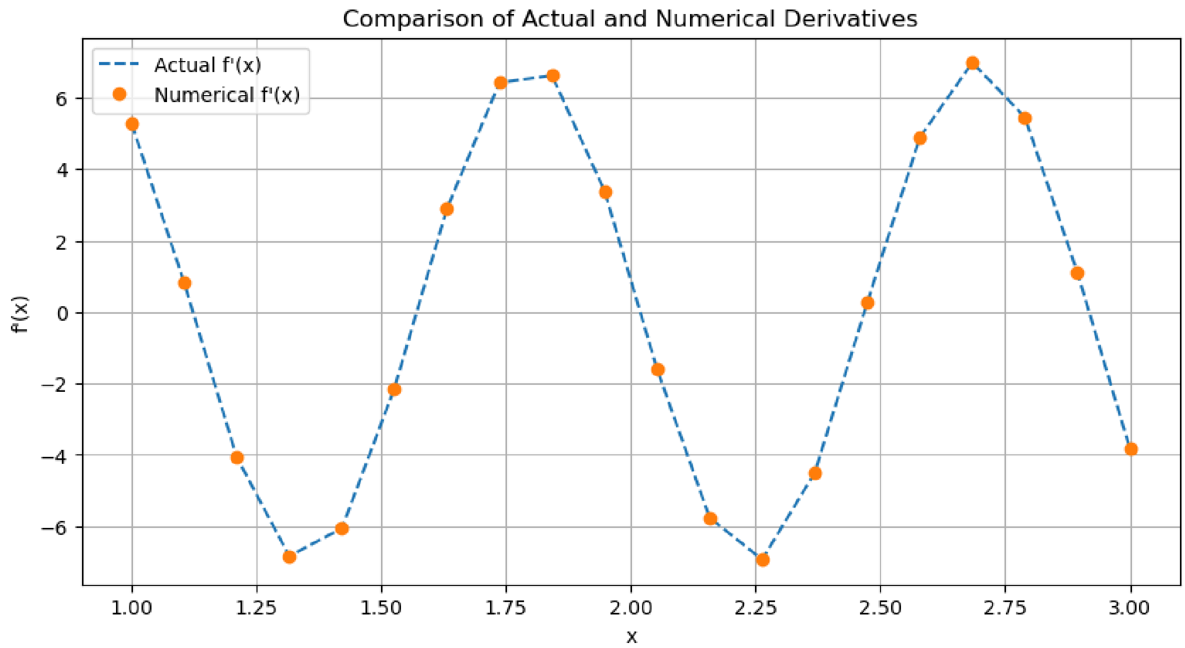


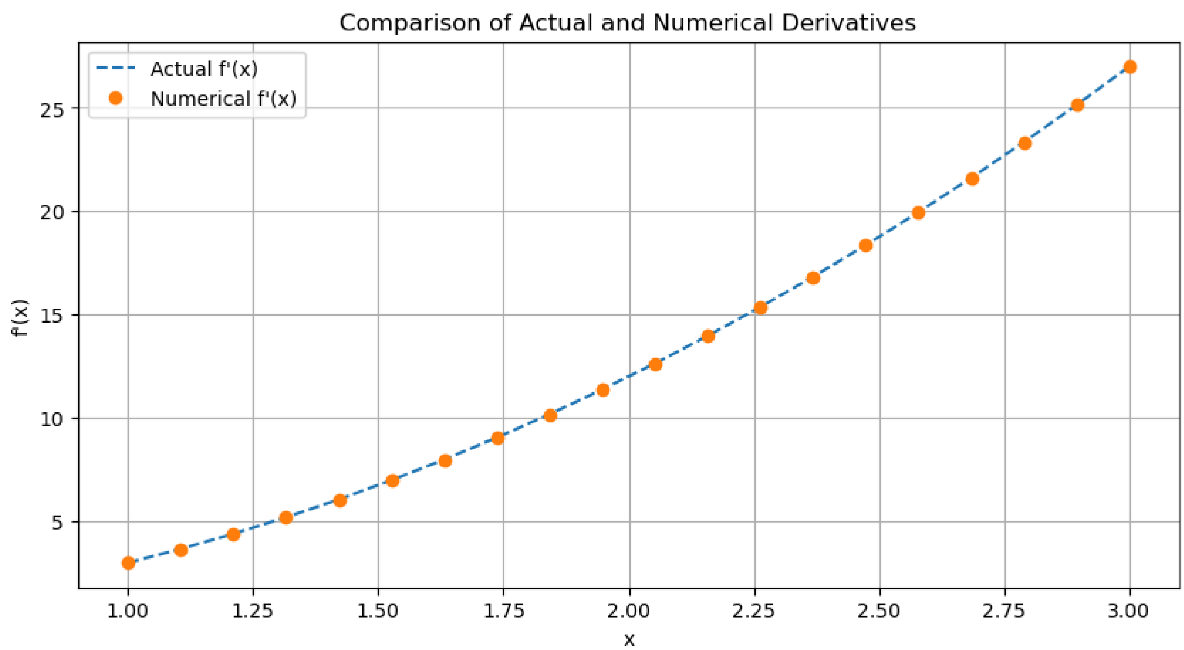
Problem 5 Looking at the graphs 20 points seems to be a reasonable amount in determining the derivative and creating a smooth graph. However functions that have a faster changing derivative may need more steps as the graph does not appear as smooth. For example, the graph of df2 below is more choppy and less smooth than the other derivative graphs.

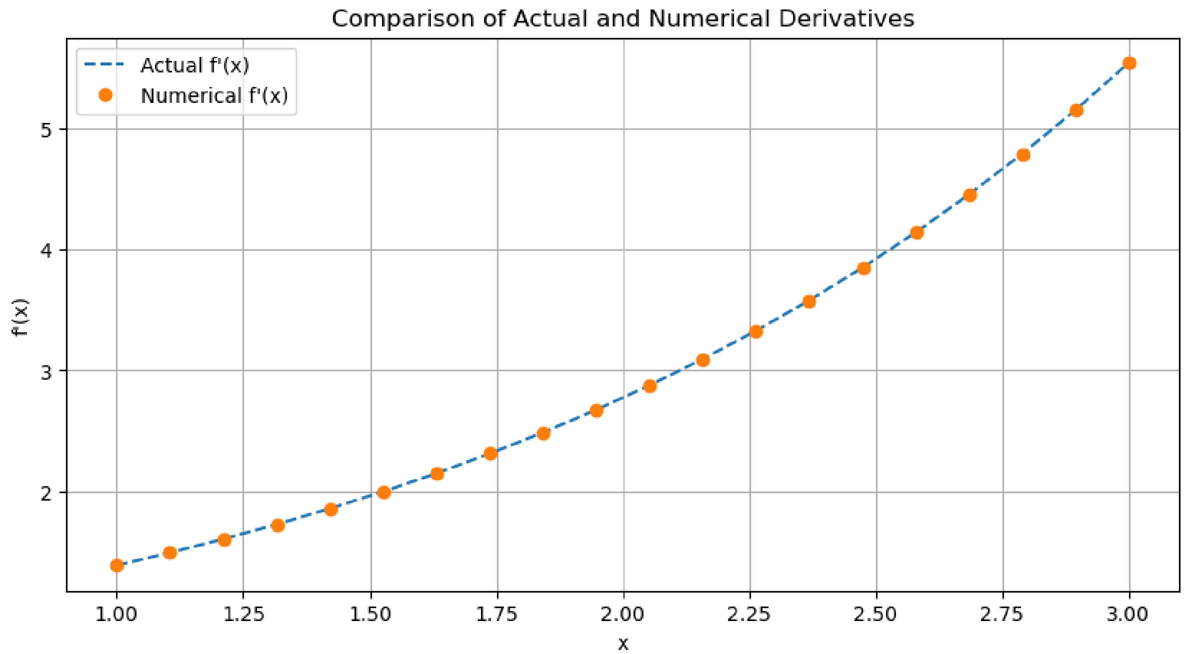
In [28]:
```
def f1(x):
    return 1/x

def df1(x):
    return -1/x**2

def f2(x):
    return np.sin(7*x)

def df2(x):
    return 7*np.cos(7*x)

def f3(x):
    return x**3

def df3(x):
    return 3*x**2

def f4(x):
```

```
    return 2**x

def df4(x):
    return np.log(2) * 2**x

import math

def Numerical_Derivative(f, x):
    h = 1e-5
    derivative = (f(x + h) - f(x - h)) / (2 * h)
    return derivative

def plot_function_and_derivative(f, df, x_start, x_end, num_points):
    x_values = np.linspace(x_start, x_end, num_points)
    actual_derivative_values = [df(x) for x in x_values]
    numerical_derivative_values = [Numerical_Derivative(f, x) for x in x_values]

    plt.figure(figsize=(10, 5))
    plt.plot(x_values, actual_derivative_values, label='Actual f\'(x)', linestyle='
    plt.plot(x_values, numerical_derivative_values, label='Numerical f\'(x)', marke
    plt.xlabel('x')
    plt.ylabel("f'(x)")
    plt.title(f'Comparison of Actual and Numerical Derivatives')
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot derivatives for each function
plot_function_and_derivative(f1, df1, 1, 3, 20)
plot_function_and_derivative(f2, df2, 1, 3, 20)
plot_function_and_derivative(f3, df3, 1, 3, 20)
plot_function_and_derivative(f4, df4, 1, 3, 20)
```




Comparison of Actual and Numerical Derivatives
Actual f'(x)
Numerical f'(x)
f'(x)
x
6
4
2
0
−2
−4
−6
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Comparison of Actual and Numerical Derivatives
Actual f'(x)
Numerical f'(x)
f'(x)
x
25
20
15
10
5
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Comparison of Actual and Numerical Derivatives
Actual f'(x)
Numerical f'(x)
f'(x)
x
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
2
3
4
5

In [ ]: