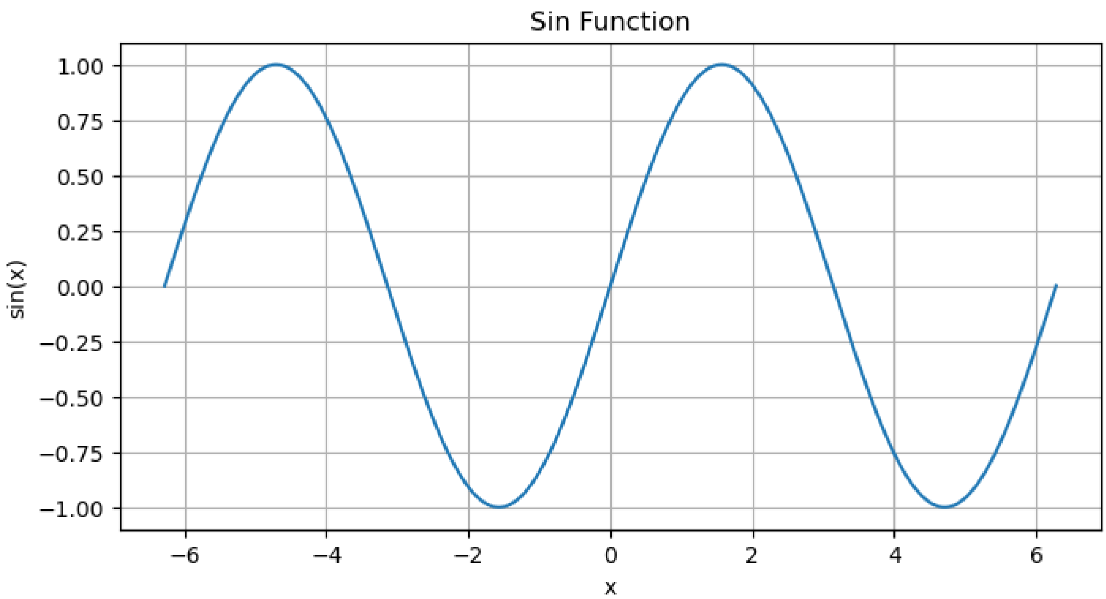
week-2-submission-1

January 29, 2024

[2]:
```python
# Imports
import matplotlib.pyplot as plt
import numpy as np
```

0.0.1 Plot Sine Function (for my reference)

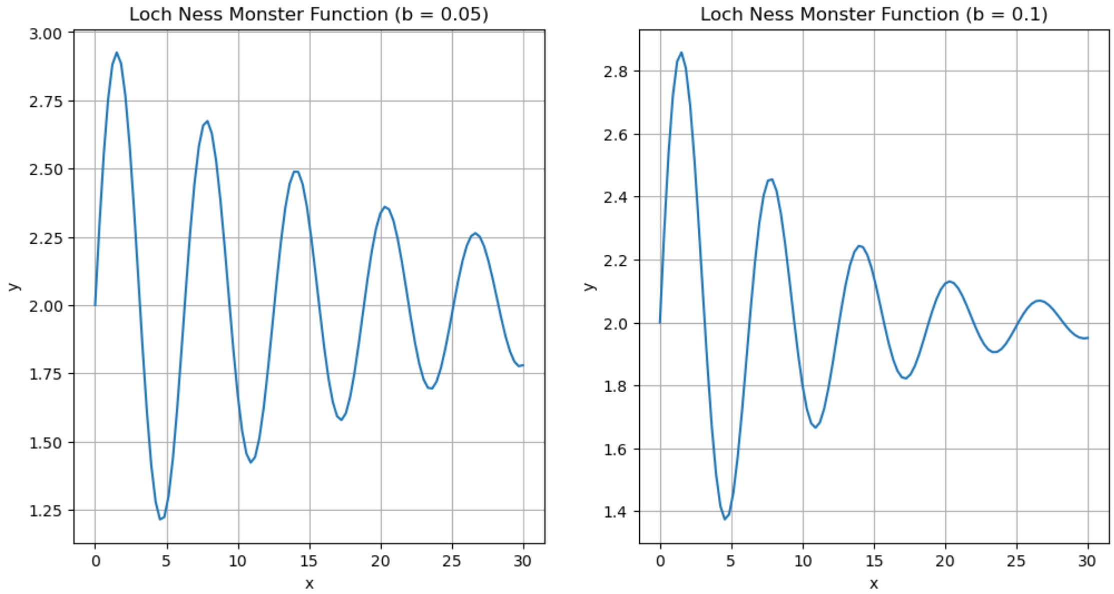
[4]:
```python
# Define the range of values for x
    # np.linesapce creates array of 1000 points evenly spaced between -2  and 2
        # This range is chosen since it covers two full cycles of the sine wave
            # sine waves repeat every 2
    # np.pi is constant for the value of   (pi).
x = np.linspace(-2*np.pi, 2*np.pi, 1000)

# Compute the sine of these values
    # Computes the sine of all the 1000 points in x and stores the result in y.
y = np.sin(x)

# Create the plot
    # Set the size
plt.figure(figsize=(8, 4))

    # Set the variables to plot
plt.plot(x, y)

    # Title and Labels
plt.title('Sin Function')
plt.xlabel('x')
plt.ylabel('sin(x)')

    # Add lines to the plot
plt.grid(True)

    # Show the plot
plt.show()
```



0.0.2 Plot Loch Ness Monster Function

[6]:
```
# Define the range of values for x
    # This range is chosen to observe the behavior of the function from 0 to 30
x = np.linspace(0, 30, 100)

# Compute the Loch Ness Monster function for b = 0.05
    # The function is y = 2 + e^(-bx)sin(x), where b is 0.05
        # e^(-bx) represents an exponential decay and sin(x) is a sine wave
    # The combination of these two creates a wave pattern that decreases in␣
 ↪amplitude over time
y1 = 2 + np.exp(-0.05 * x) * np.sin(x)

# Compute the Loch Ness Monster function for b = 0.1
    # This leads to a faster decay of the wave pattern
y2 = 2 + np.exp(-0.1 * x) * np.sin(x)

# Create the plot
plt.figure(figsize=(12, 6))

# Plot for b = 0.05
    # Create the subplot on the left
plt.subplot(1, 2, 1)
plt.plot(x, y1)
plt.title('Loch Ness Monster Function (b = 0.05)')
plt.xlabel('x')
plt.ylabel('y')
```

```
plt.grid(True)

# Plot for b = 0.1
    # Create the subplot on the right
plt.subplot(1, 2, 2)
plt.plot(x, y2)
plt.title('Loch Ness Monster Function (b = 0.1)')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)

# Show the plots
plt.show()
```

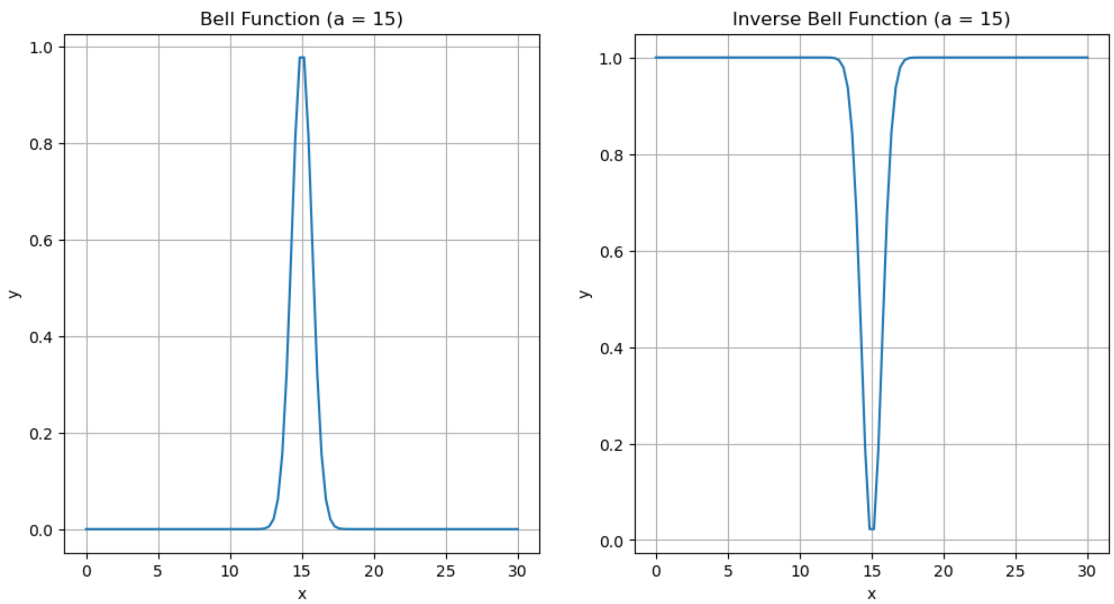


The 0.1 graph is decaying faster than the 0.05 graph since when b is larger, the value of -bx becomes more negative which in return makes e^-bx smaller. Therefore making the y smaller as well.

0.0.3 Bell Function

[9]:
```
# Define the range of values for x
x = np.linspace(0, 30, 100)

# Define the center of the Bell curve
a = 15.0

# Compute the Bell function
    # The function is defined as y = exp(-(x-a)^2)
```

```python
y_bell = np.exp(-(x - a)**2)

# Compute the Inverse Bell function
y_inverse_bell = 1 - np.exp(-(x - a)**2)

# Create the plot
plt.figure(figsize=(12, 6))

# Plot for the Bell function
plt.subplot(1, 2, 1)
plt.plot(x, y_bell)
plt.title('Bell Function (a = 15)')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)

# Plot for the Inverse Bell function
plt.subplot(1, 2, 2)
plt.plot(x, y_inverse_bell)
plt.title('Inverse Bell Function (a = 15)')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)

# Show the plots
plt.show()
```



The reason why the graphs look the way they do because it is a bell-shaped curve centered at $x = a$ (here $a = 15$). The initial function value increases as x approaches a, and decreases as x moves away from a. The inverse does the opposite so the function value decreases towards 0 as x approaches a, and increases towards 1 as x moves away from a.