Lab Assignment 30

Student Name: Chauhan Vandana Ramdayal

Student Id: AF0411629

Topic: Scipy Transform,Interpolation and IO

1. Scipy Transform

- **Purpose**: Transformation functions in Scipy are used for mathematical transformations such as Fourier transforms, which are used in signal processing, image processing, and data analysis.
- **Common Modules**:
  - **scipy.fft**: Provides functions for fast Fourier transforms (FFT), allowing you to convert signals between time and frequency domains.
    - Example: scipy.fft.fft performs a fast Fourier transform on a signal.
    - Applications: Useful in analyzing frequencies in audio signals, filtering noise, and image processing.
  - **scipy.ndimage**: Has functions for multidimensional image processing, like rotating, shifting, and resizing images.
- **Use Case**: If you want to analyze the frequency components of a sound or image, you could use FFT to transform the data.

2. Scipy Interpolation

- **Purpose**: Interpolation is used to estimate unknown values between two known values. It's commonly used in data fitting, plotting smooth curves, and filling in missing data points.
- **Common Functions**:
  - **scipy.interpolate.interp1d**: For 1-dimensional linear interpolation. It creates a function that interpolates between the points you provide.
  - **scipy.interpolate.griddata**: For multi-dimensional data interpolation. It interpolates irregularly spaced data points onto a regular grid.
  - **Interpolation Types**:
    - **Linear**: Straight-line estimation between points.
    - **Quadratic & Cubic**: More complex, curved estimation methods that provide a smoother curve.
  -

3. Scipy IO (Input/Output)

- **Purpose**: Scipy provides tools to read and write various types of data files, including .mat files, which are commonly used with MATLAB.
- **Common Functions**:
  - **scipy.io.loadmat**: Loads data from a .mat file, making it easy to transfer data between MATLAB and Python.
  - **scipy.io.savemat**: Saves data to a .mat file.
  - **scipy.io.wavfile**: Reads and writes .wav files (audio files). It can be used to process audio data directly in Python.

Q1.To Find estimate temperature given known data points by using interpolation:

Input -

time_data =[0, 1, 2, 3, 4, 5, 6])

temperature_data=[20, 22, 24, 23, 21, 18, 15]

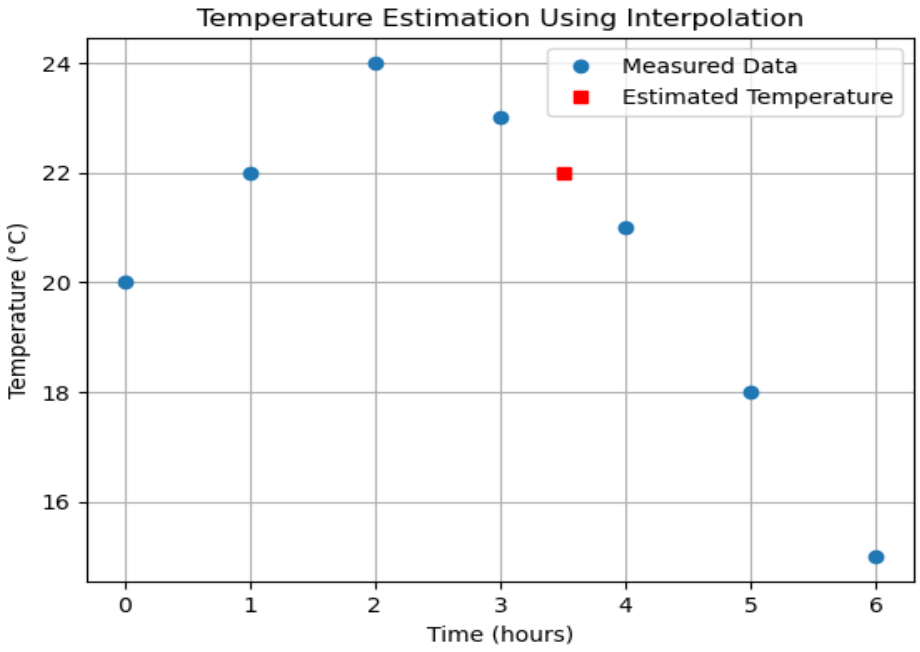
Code:

```
from scipy.interpolate import interp1d
import numpy as np
import matplotlib.pyplot as plt
time_data = np.array([0, 1, 2, 3, 4, 5, 6])
temperature_data = np.array([20, 22, 24, 23, 21, 18, 15])

interpolation_function = interp1d(time_data, temperature_data, kind='linear') # Create an interpolation function using linear interpolation
desired_time = 3.5  # Time at which you want to estimate the temperature
estimated_temp = interpolation_function(desired_time) # Use the interpolation function to estimate the temperature

# Plot the known data and the estimated temperature
plt.plot(time_data, temperature_data, 'o', label='Measured Data')
plt.plot(desired_time, estimated_temp, 's', label='Estimated Temperature', color='red')
plt.xlabel('Time (hours)')
plt.ylabel('Temperature (°C)')
plt.title('Temperature Estimation Using Interpolation')
plt.legend()
plt.grid()
plt.show()
print(f"Estimated Temperature at {desired_time} hours: {estimated_temp:.2f} °C")
```

Output:




```
Estimated Temperature at 3.5 hours: 22.00 °C
```

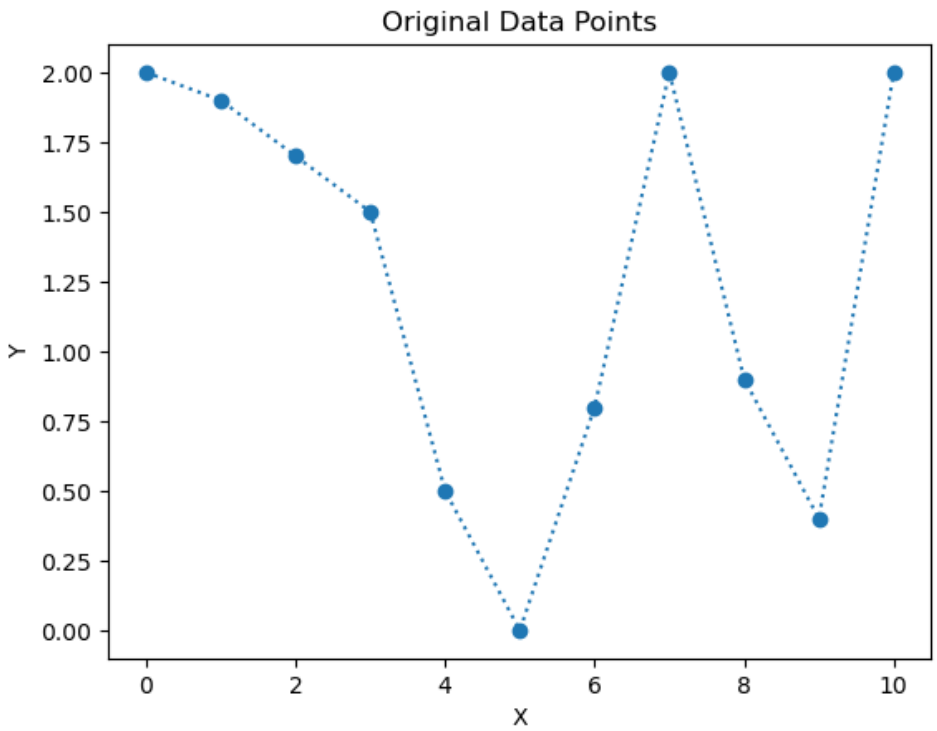
Q2. To Find estimate values range 1 to 100 with known data values

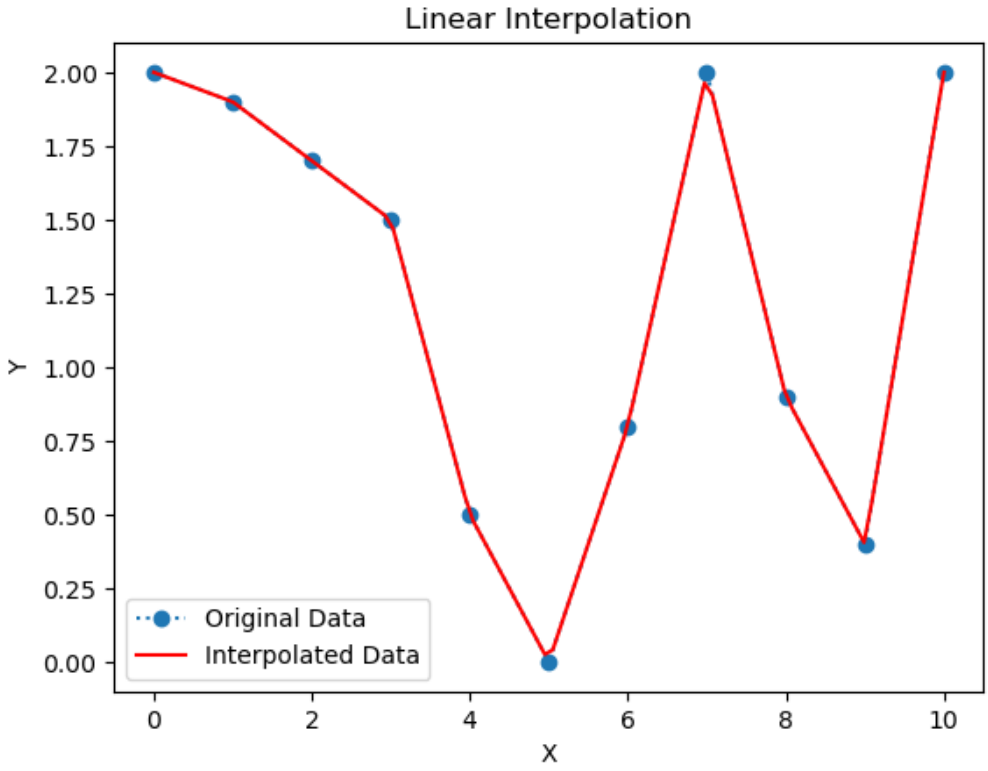
Code:

```
from scipy.interpolate import interp1d
import numpy as np
import matplotlib.pyplot as plt
# Define known data points
X = np.arange(11)  # X values from 0 to 10
print("X:", X)
Y = np.array([2.0, 1.9, 1.7, 1.5, 0.5, 0.0, 0.8, 2.0, 0.9, 0.4, 2.0])
print("Y:", Y)
plt.plot(X, Y, 'o:', label="Original Data") # Plot the original data points
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Original Data Points")
plt.show()
predict = interp1d(X, Y, kind='linear') # Set up the interpolation function
X2 = np.linspace(0, 10, 100)
Y2 = predict(X2)  # Interpolated Y values
# Plot the interpolated values
plt.plot(X, Y, 'o:', label="Original Data")
plt.plot(X2, Y2, 'r-', label="Interpolated Data")
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear Interpolation")
plt.legend()
plt.show()
```

Output:

```
X: [ 0  1  2  3  4  5  6  7  8  9 10]
Y: [2.  1.9 1.7 1.5 0.5 0.  0.8 2.  0.9 0.4 2. ]
```




Linear Interpolation
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
0
2
4
6
8
10
X
Original Data
Interpolated Data