**Lab 5: NumPy and Basic Plotting**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: | Shaheer Ziya |  | University Number: | 3035946760 |

**Exercise 1: Chebyshev Polynomial of the First Kind**

AIM:

Write a Python program that uses the NumPy Polynomial class to print a table of the first ten Chebyshev polynomials of the first kind. Here is the table generated by this program:

T\_0(x) = 1

T\_1(x) = x

T\_2(x) = 2\*x^2 - 1

ALGORITHM:

|  |
| --- |
| * Obtain the coefficients for the first ten Chebyshev polynomials and store them in a tuple * Loop over each list containing the coefficients for that order Chebyshev polynomial in reverse and adjust the coefficients for printing in descending order * Append the coefficients to a polynomial string that is printed at the end of the iteration |

PROGRAM:

|  |
| --- |
| # Chebyshev Polynomials  # Created by Shaheer Ziya  import numpy as np  def main():    coeff\_table = tuple()    chebyshev\_coefficents = [1]  for i in range(10):  coeff\_table += list(np.polynomial.Chebyshev.convert(np.polynomial.Chebyshev(coef=chebyshev\_coefficents),  kind = np.polynomial.Polynomial).coef),  chebyshev\_coefficents.insert(0, 0)      # Print the polynomial  for i, pol in enumerate(coeff\_table):  polynomial\_str = ""  for idx, coeff in enumerate(pol[::-1]):  # Determine the sign of the coefficient  if coeff > 0: sign = " + "  else: sign = " - "    # Determine the power of the coefficient  power = len(pol) - idx - 1  # Determine the string representation of the coefficient  if (coeff == 1 and power != 0): coeff\_str = ""  else: coeff\_str = str(abs(int(coeff)))    if coeff == 0:  continue  elif power == 0:  polynomial\_str += sign + coeff\_str  elif power == 1:  polynomial\_str += sign + coeff\_str + "x"  else:  polynomial\_str += sign + coeff\_str + "x^" + str(power)    print(f"T\_{i}(x) = " + polynomial\_str[3:])  main() |

OUTPUT:

|  |
| --- |
| Text  Description automatically generated |

**Exercise 2: Sound Intensity from a Point Source**

AIM:

In an experiment, Mary measured the variation of the intensity *I* of the sound produced by a point source with the distance *r* from the source. Here is her measurement result:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *r* (m) | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.0 | 2.2 | 2.4 |
| *I* (10−5 W/m2) | 0.987 | 0.662 | 0.525 | 0.373 | 0.308 | 0.262 | 0.191 | 0.184 |

On the other hand, physical theories tell us that for a point source of sound of power *P*, the sound intensity *I* and the distance *r* from the source are related by:

Write a Python program that uses the np.linalg method lstsq to find the best least-square fit of

for the given data and then display the fitting result together with the theoretical prediction. Your program should output a table of the values of the fitting parameters *m* and *k* found from the fitting and their theoretical values as well as the root-mean-square of the residual of the fitting. Assume that the point source emits sound with a power of

ALGORITHM:

|  |
| --- |
| * Feed the values for the data (r and I) * Prepare the data to be modelled using lstq * Fit it with np.linalg.lstq() * Print the fitted values for m, k and residue * Print both the fitted data and the theoretical redictions |

PROGRAM:

|  |
| --- |
| # Data Fit  # Created by Shaheer Ziya  import matplotlib.pyplot as plt  import numpy as np  P = 4 \* np.pi \* 10e-5  # Radii & Intensity  r = np.array([1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4])  I = np.array([0.987,0.662,0.525,0.373,0.308,0.262,0.191,0.184]) \* 10e-5  def main():  A = np.vstack((np.log(r), np.ones(len(r)))).T  B = np.log(I)  x, resid = np.linalg.lstsq(A, B)[0:2]  m, k = x  print(m, k, resid)  plt.subplots()  # plt.plot(np.log(r), np.log(I), 'o', label='Data')  plt.plot(np.log(r), np.log(P / (4 \* np.pi \* r\*\*2)), 'o', label='Theory')    plt.plot(np.log(r), m\*np.log(r) + k, 'r', label='Fit')  plt.title("Sound Intensity from a Point Source")  plt.xlabel("Log(r) (m)")  plt.ylabel("Log(I) $(W/m^{-2})$")  plt.show()  main() |

OUTPUT:

|  |
| --- |
| Chart, line chart  Description automatically generated |

**Exercise 3: Forced Vibration with Damping**

AIM:

A small block of mass *m* suspended vertically by a spring with spring constant *k* is driven by an external force *F*(*t*) = *F*0 cos(ω*t*). The block is moving in a viscous medium with a damping force of the form −*bv* where *b* > 0 is the damping constant and *v* is its instantaneous velocity. Taking downward as the positive direction, the vibration of the block is modeled by the differential equation:

where *x*(*t*) is the displacement of the block from its equilibrium position at time *t*. It can be shown that the steady state solution (i. e. *x*(*t*) when time *t* → ∞) is

In this formula, *M* is the magnification ratio and *φ* is the phase lag defined by

where is the natural frequency and is the damping ratio. Write a Python program that uses the Matplotlib Axes class method plot to plot the magnification ratio *M* over the interval of frequency ratio ω/ω0 from 0 to 2.0 for damping ratio 0.1, 0.2, 0.4, 0.6, and 0.8, respectively, on the same graph. You should label your graph with proper axis labels, title, and legends. From your graph, you can observe how the peak value of *M* depends on , i. e. the effect of damping on the resonance frequency of the block.

ALGORITHM:

|  |
| --- |
| * Prepare the data * Loop over each xi value and find the magnification ratio and phase-lag, then plot it on the subplot * Add titles and show legend |

PROGRAM:

|  |
| --- |
| # Plot Damped Oscillator  # Created by Shaheer Ziya  import matplotlib.pyplot as plt  import numpy as np  wOverw0 = np.linspace(0, 2, 1000)  xiS = np.array([0.1, 0.2, 0.4, 0.6, 0.8])  def main():    plt.subplots()    for xi in xiS:  M = 1 / (np.sqrt(  ((1 - wOverw0\*\*2)\*\*2) + (4 \* xi\*\*2 \* wOverw0\*\*2)  ))  phi = np.arctan2(2\*xi\*wOverw0, 1 - wOverw0\*\*2)  plt.plot(phi, M, label=f"{xi}")    plt.legend()  plt.xlabel("Phase Lag $\phi$")  plt.ylabel("Maginification ratio")  plt.title("Forced Vibration with Damping")    plt.show()  main() |

OUTPUT:

|  |
| --- |
| Chart  Description automatically generated |

**Exercise 4: Employees in Hong Kong's Construction Industry**

AIM:

Below is the table of the employment statistics in Hong Kong's construction industry from 2011 to 2020 (source: https://www.censtatd.gov.hk/tc/scode200.html by Census and Statistics Department, HKSAR).

|  |  |  |
| --- | --- | --- |
| Year | Number of Employees in Thousands | Share of the Employees in the Labour Force |
| 2011 | 277.0 | 7.75 % |
| 2012 | 290.1 | 7.93 % |
| 2013 | 309.0 | 8.30 % |
| 2014 | 309.7 | 8.27 % |
| 2015 | 316.7 | 8.39 % |
| 2016 | 328.4 | 8.67 % |
| 2017 | 342.0 | 8.95 % |
| 2018 | 351.6 | 9.09 % |
| 2019 | 337.5 | 8.77 % |
| 2020 | 310.0 | 8.47 % |

Write a Python program that uses Matplotlib Axes class method twinx to produce a bar chart of the number of employees in Hong Kong’s construction industry and a line plot of the percentage share of these employees in the labour force as a function of year on the same graph. You should label your graph with proper axis labels, title, and legends.

ALGORITHM:

|  |
| --- |
| * Initialize the data to be printed * Plot the first set of data on axs1, labelling the data as we go along * Instantiate the other twin axis * Plot its data, labelling the axis and showing its legend * Show the final plot |

PROGRAM:

|  |
| --- |
| # Twin Bar Chart  # Created by Shaheer Ziya  import matplotlib.pyplot as plt  import numpy as np  years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]  numEmployees = [277.0, 290.1, 309.0, 309.7, 316.7, 328.4, 342.0, 351.6, 337.5, 310.0]  shareEmployees = [7.75, 7.93, 8.30, 8.27, 8.39, 8.67, 8.95, 9.09, 8.77, 8.47]  def main():  fig, ax1 = plt.subplots()  ax1.bar(years, numEmployees, label = 'Number of Employees')  ax1.set\_ylabel('Number of Employees in Thousands')  ax2 = ax1.twinx()  ax2.plot(years, shareEmployees, label = 'Share of Employees', color = 'r')  ax2.set\_ylabel('Share of Employees')  ax2.set\_yticklabels(['{}%'.format(x) for x in shareEmployees])  ax1.legend(loc='upper left')  ax2.legend(loc='lower right')  plt.title('Employees in Hong Kong\'s Construction Industry')  plt.show()  main() |

OUTPUT:

|  |
| --- |
| Chart, line chart, histogram  Description automatically generated |