S16036

CPL101

1. Interpolation vs Regression

I. Regression is a statistical method used to understand the relationship between variables and it is used to find the relationship between one dependent variable and one or more independent variable.
Interpolation means a method which is used to find unknown values that fall in between known values.

II.

Advantages	Disadvantages
• Helps to determine unknown data points which are in between known data points	• Error is proportional to the square of the distance between the two graphs
• Precise	• Not precise
• Easy to use as it does not require extensive computational resources	• Assumptions are not always correct
• Flexible	• Predictions which are made out of the data range may be unreliable

III.

Advantages	Disadvantages
• Easy to understand	• Relies on assumptions strongly.
• Shows the strength between variables	• Sensitive to outliers
• Can be used in various fields	• Can be used only with few numbers of variables
• Can interpret easily and directly	• Specifying the model incorrectly

IV. Regression
Reason: Regression is the best method as projecting y values between the observed time interval since it can manage extrapolation to estimate future values

V. The predicted model using the interpolation method can be fitted to noise and outliers present in the dataset. It impacts the correctness of the data model. Inaccurate models may produce unreliable forecasted data points.

VI. Noise and outliers may affect strongly as they are used to determine the best fitted line using regression. Noises has an impact on the correlation between the data points and outliers will alter the pattern of the data set. So the model may be overfitted or underfitted.

2. Least Squares Regression

I.

a) Least Squares regression is a method which is used to find the best fit line of a set of data. Its main primary objective is to minimize the sum of offset or residuals from the given set of data and then predict the relationship between the variables.

b) It minimizes the sum of squared vertical distances between the fitting line and the data points. Its primary goal is to reduce the difference between residuals and observed values and fit the line accordingly.

c) Finding slope: Initially find the values of x^2 and xy and after that obtain the sum of the values of $\sum(xy), \sum x^2, \sum x, \sum y, \sum xy$. Substitute those values to the following equation and find the gradient.

$$m = \frac{N\sum(xy) - \sum x \sum y}{N\sum(x^2) - (\sum x)^2}$$

Find interception: Substitute values for the following equation as required and do the calculations.

$$b = \frac{\sum y - m\sum x}{N}$$

d) The regression model has linearity in its error term and coefficients.
The error term's population mean is zero.
No correlation between the independent variable and the error term.
Each observation of the error term is independent of others.
The error terms' variance is constant.
The error term adheres to a normal distribution pattern.

e) The residual sum of squares gives tells how well the regression model fits the data. It calculates the residuals of the model and the variance of errors. If residual sum of squares is smaller, then the data points will fit the model better.

f) Mean absolute error means the average absolute difference between the observed values and the predicted values. It tells how large the errors are from the average.

$$Mean\ absolute\ error = \frac{1}{n}\sum_{i=1}^{n}|y_i - y'_i|$$

n = number of observations

y_i = observed values

y'_i = predicted values

g) Initially the predicted values must be generated using for each observation using the regression model given below.

$y'_i = mx_i + b$

m = gradient

b = interception

Then find the absolute error for each observation using the following equation.

$$|e_i| = |y_i - y'_i|$$

$$e_i = error$$

$$y_i = observed\ value$$

$$y'_i = predicted\ value$$

Finally mean absolute error is found using the following equation.

$$Mean\ absolute\ error = \frac{1}{n}\sum_{i=1}^{n}|y_i - y'_i|$$

If the mean absolute error is lower then better the model fits the data and more accurate.

II.

a) $T^2 = \left(\frac{4\pi^2}{k}\right) m$

$T^2 = y$

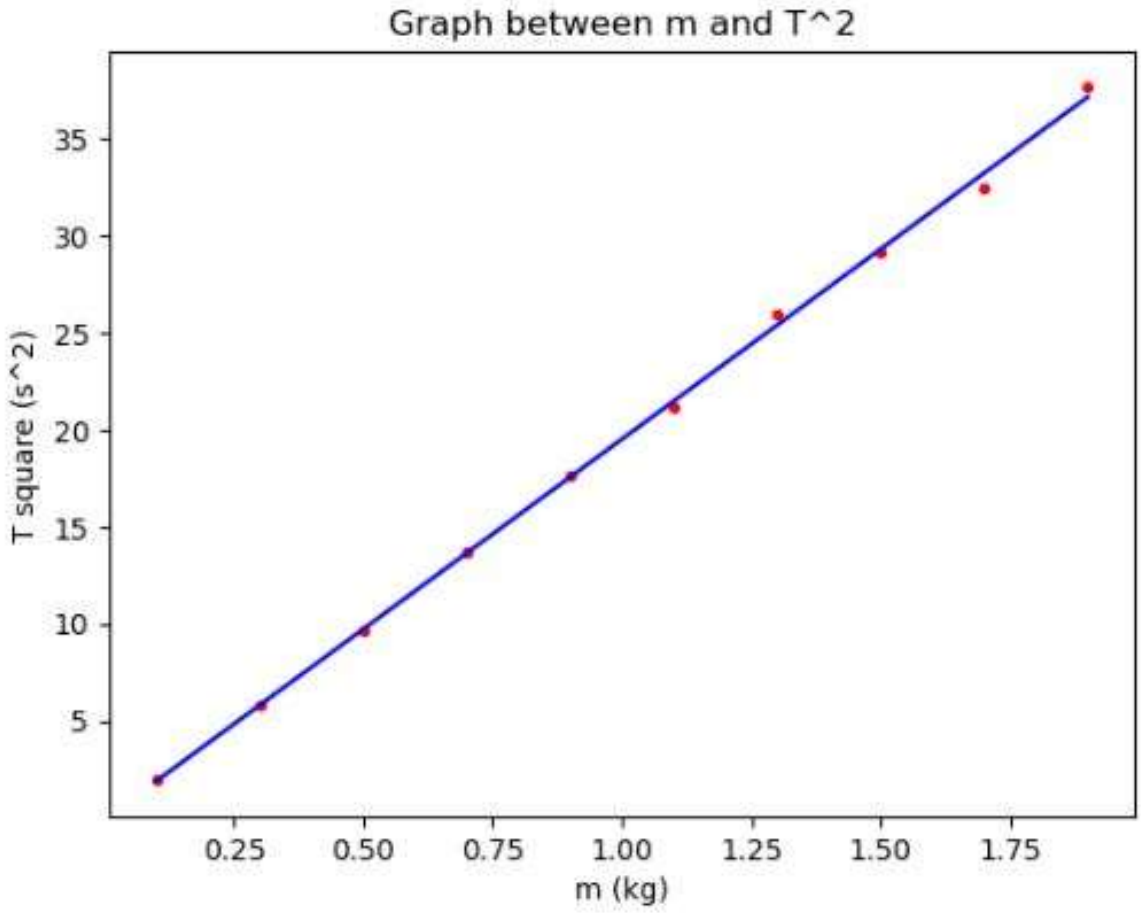
$m = x$

$\frac{4\pi^2}{k} = b$

b)

```
import numpy as np

import math

import matplotlib.pyplot as plt


m = np.array([0.1,0.3,0.5,0.7,0.9,1.1,1.3,1.5,1.7,1.9])

T = np.array([1.4,2.4,3.1,3.7,4.2,4.6,5.1,5.4,5.7,6.14])

T_square = np.square(T)


matrixA = np.vstack([m, np.ones(len(m))]).T

T_square = T_square[:, np.newaxis]


alpha = np.dot((np.dot(np.linalg.inv(np.dot(matrixA.T,matrixA)),matrixA.T)),T_square) #least square regression

print(alpha)


plt.figure()

plt.plot(m,T_square,'r.') #to plot the data points

plt.plot(m, alpha[0]*m + alpha[1], 'blue') #to plot the graph
```

```
plt.title('Graph between m and T^2')
plt.xlabel("m (kg)")
plt.ylabel("T square (s^2)")
plt.show()

k = (4*math.pi**2)/alpha[0]
print("spin constant(k) = ",k)
```




```
spin constant(k) =  [2.01359355]
```

3. Polynomial Regression

I.

a) In polynomial regression it gives a curved relationship between independent and dependent variables which gets fit to the data more accurately. Furthermore, it can be modeled as an n^{th} degree polynomial also.
Least square regression is a simple linear relationship in the form of y = mx + b and it can be used to reduce the variance between two variables.

b) Fitting a polynomial function to the observed data points allows polynomial regression to model the relationship between variables. The coefficients of the polynomial function are estimated so that it fits the data. Least squares are used for it.

c) The highest power of the independent variable that appears in the polynomial is called as the degree of a polynomial.

It represents the flexibility and complexity of he model and it helps to find the relationship between independent and dependent variables. If the appropriate degree is found, we can reduce the complexity and overfitting of the model.

d) Initially the polynomial equation is formulated according to the increasing order of the degrees of x. Then a design matrix(X) is constructed so that the columns represent the powers of the independent variable. Then a response vector(y) is constructed so that it contains the observed values of independent variables. Finally the coefficients are estimated using the following equation.

$$b = (X^T X)^{-1} X^T y$$

B – vector of coefficients

X^T – Transporse of design matrix X

y – response vector

e) A popular statistic for assessing a regression model's accuracy is mean squared error. It measures the discrepancies between the values that are seen and the values that the model predicts, or the average of the squares of the errors. Due to its ability to

produce a single number that accurately represents the overall performance of the model.

$$Mean\ sqared\ error = \frac{1}{n}\sum_{n=1}^{n}(y_i - y'_i)^2$$

n = number of observations

y_i = observed values

y'_i = predicted values

f) First the polynomial regression model must be fitted to the training data as follows. (d is the degree of polynomial regression model.)

$$y' = b_0 + b_1x + b_2x^2 + \ldots + b_dx_i^d$$

Next the predicted values are found using the observed values.

Residuals are obtained next by subtracting predicted values from the observed values.

Mean squared error is calculated using the following equation.

$$Mean\ sqared\ error = \frac{1}{n}\sum_{n=1}^{n}(y_i - y'_i)^2$$

n = number of observations

y_i = observed values

y'_i = predicted values

g) Impact on model performance:
If we select a low degree polynomial, the patterns in the data could not be identified as the model is being too simple and the variance also decreases. As the model fails the performance is also very low.
If we select a higher degree polynomial, the model gets more flexible as its less bias. As the model captures noise, the variance is high and due to overfitting, it has poor performance on the test dataset.

If the degree is optimal, the bias and variance balances. The variance is also moderate, and a good performance can be observed as the model captures the best trend ongoing.

Impact on complexity:
In low degree polynomials, the complexity is low as it has only few parameters. It has high interpretability and less computational cost as there are no complex simplifications.
In high degree polynomials, since the number of parameters is high, the complexity is high and therefore it has low interpretability. As there are many complex calculations, the computational cost is very high.

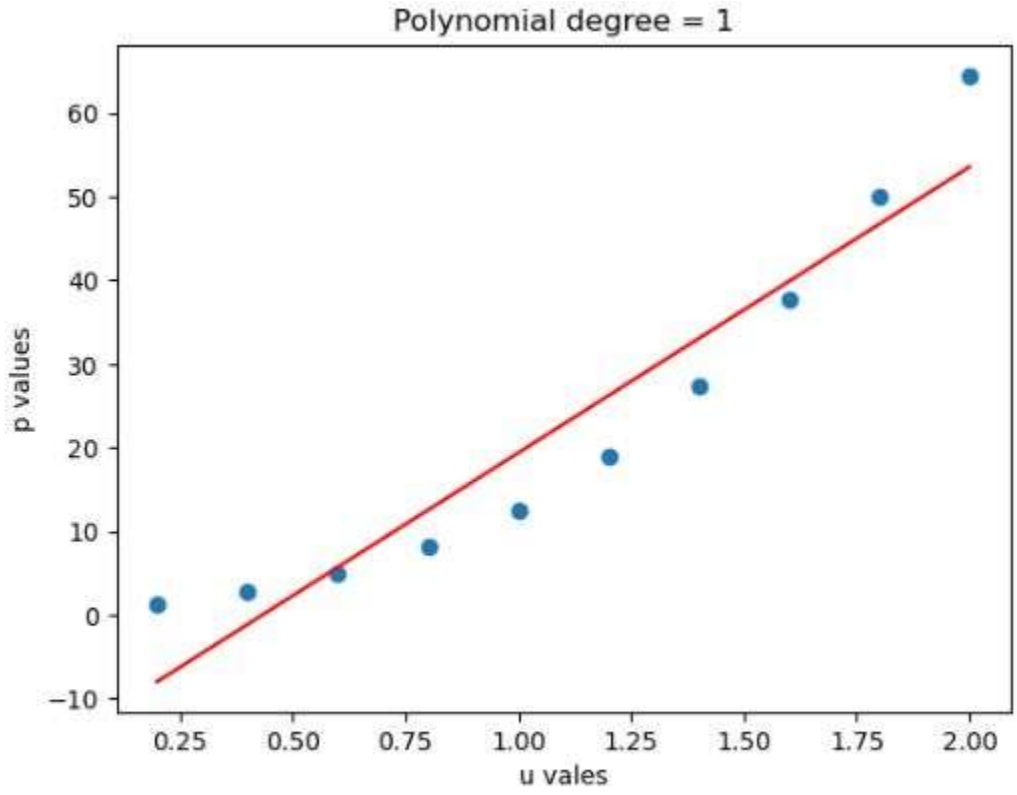
h) Polynomial regression becomes overfitted when the model captures the underlying relationship between the variables and the noise in the training data. So overfitting of polynomial expression can be addressed from regularization, cross validation etc.

II)

a)

```
# part a
import numpy as np
import math
import matplotlib.pyplot as plt

u = np.array([0.2,0.4,0.6,0.8,1.0,1.2,1.4,1.6,1.8,2.0])
p = np.array([1.3,2.7,4.9,8.1,12.5,18.9,27.3,37.7,50.1,64.5])

A = np.polyfit(u,p,1)
plt.plot(u,p,'o')
B = np.polyval(A,u)
plt.plot(u,B,'r')
plt.title('Polynomial degree = 1')
```

```
plt.xlabel('velocity (ms^-1)')
plt.ylabel('momentum (Kgms^-1)')
plt.show()

print("This underfits. Therefore, 1 is not suitable as the polynomial order")
```



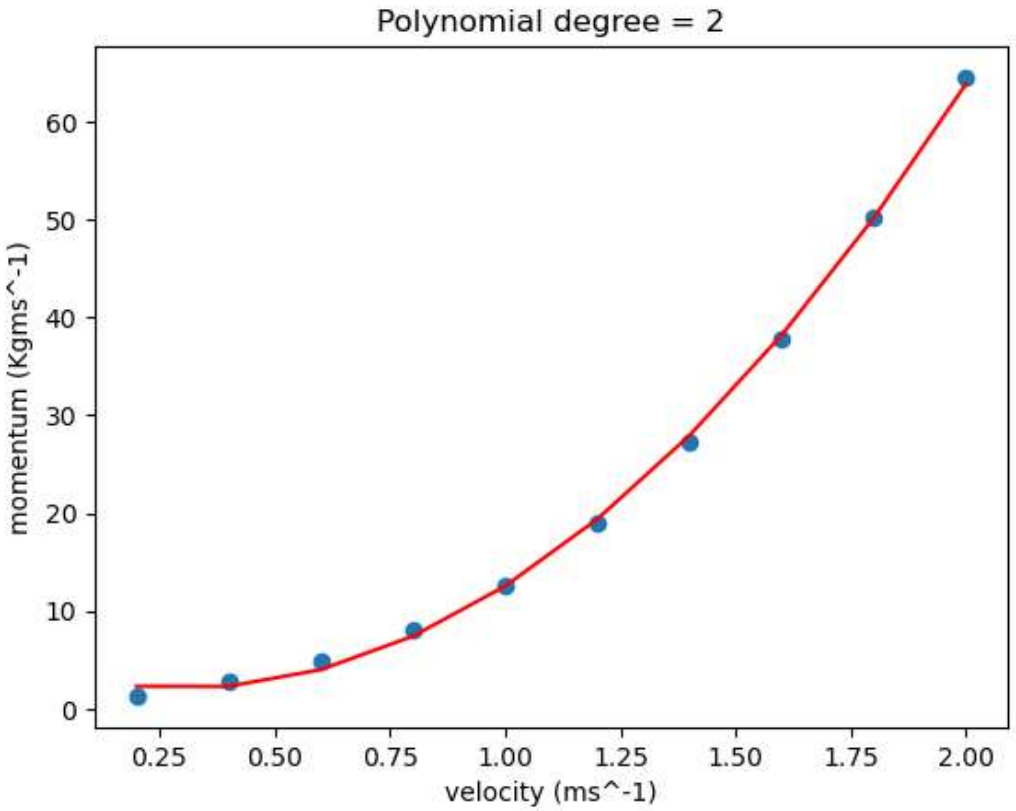

```
This underfits. Therefore, 1 is not suitable as the polynomial order
```

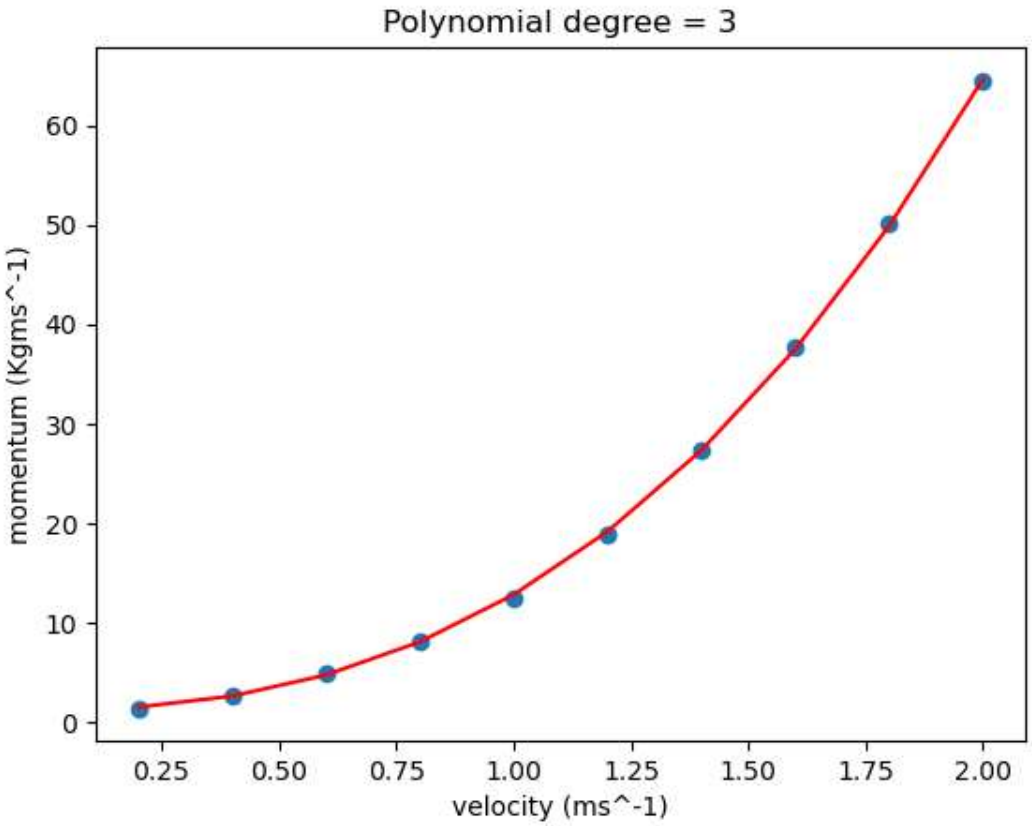
b)

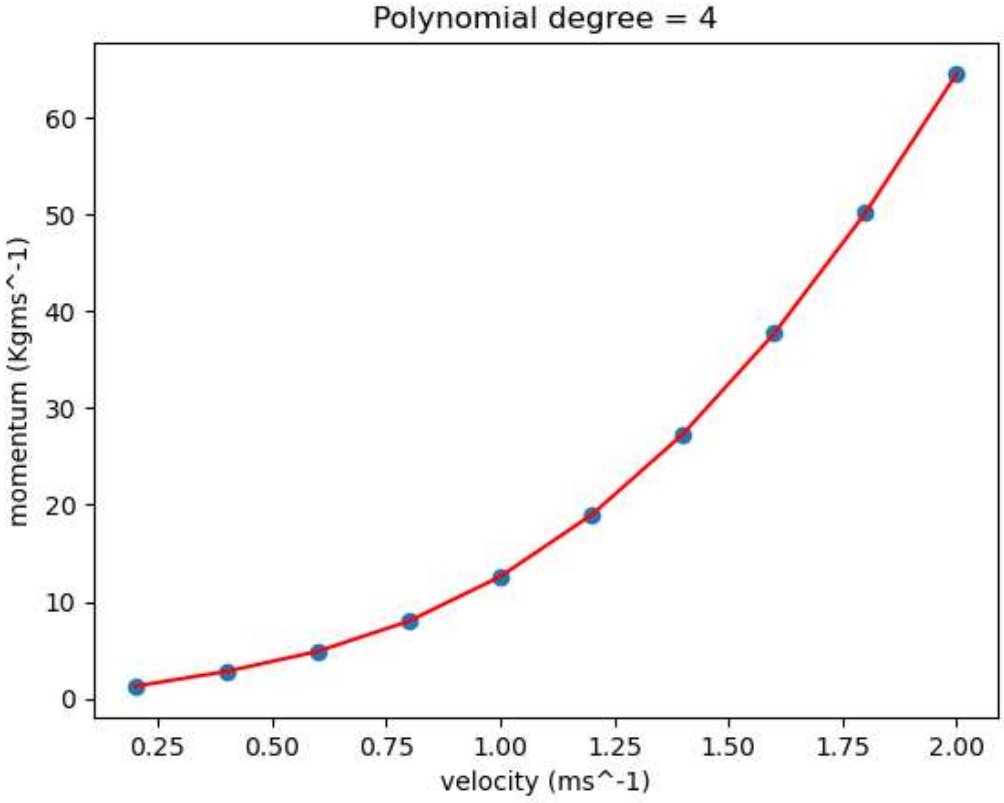
```
# part b
import numpy as np
import math
import matplotlib.pyplot as plt

u = np.array([0.2,0.4,0.6,0.8,1.0,1.2,1.4,1.6,1.8,2.0])
```

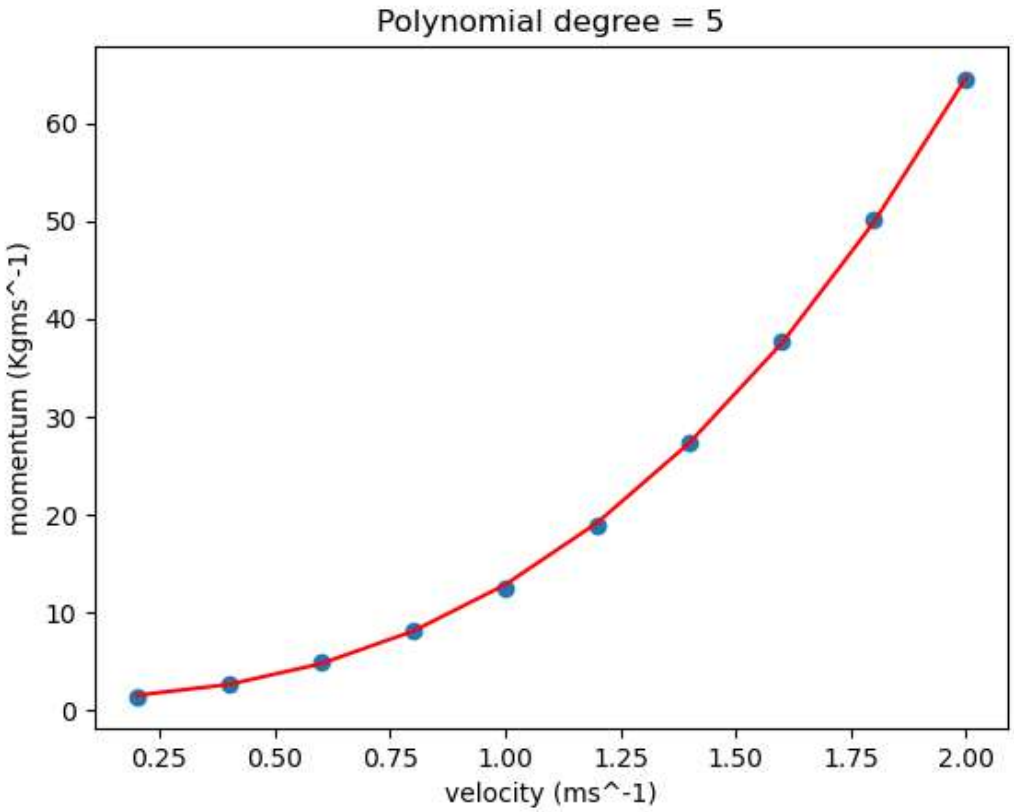
```
p = np.array([1.3,2.7,4.9,8.1,12.5,18.9,27.3,37.7,50.1,64.5])

C = np.polyfit(u,p,2)
plt.plot(u,p,'o')
D = np.polyval(C,u)
plt.plot(u,D,'r')
plt.title('Polynomial degree = 2')
plt.xlabel('velocity (ms^-1)')
plt.ylabel('momentum (Kgms^-1)')
plt.show()

E = np.polyfit(u,p,3)
plt.plot(u,p,'o')
F = np.polyval(E,u)
plt.plot(u,F,'r')
plt.title('Polynomial degree = 3')
plt.xlabel('velocity (ms^-1)')
plt.ylabel('momentum (Kgms^-1)')
plt.show()

G = np.polyfit(u,p,4)
plt.plot(u,p,'o')
H = np.polyval(G,u)
plt.plot(u,H,'r')
plt.title('Polynomial degree = 4')
plt.xlabel('velocity (ms^-1)')
plt.ylabel('momentum (Kgms^-1)')
```

```
plt.show()


I = np.polyfit(u,p,3)
plt.plot(u,p,'o')
J = np.polyval(I,u)
plt.plot(u,J,'r')
plt.title('Polynomial degree = 5')
plt.xlabel('velocity (ms^-1)')
plt.ylabel('momentum (Kgms^-1)')
plt.show()
```




Polynomial degree = 3
momentum (Kgms^-1)
velocity (ms^-1)
0
10
20
30
40
50
60
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Polynomial degree = 4
momentum (Kgms^-1)
velocity (ms^-1)
0
10
20
30
40
50
60
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



c)

```
# part c
import numpy as np
import math
import matplotlib.pyplot as plt

I = np.polyfit(u,p,5)
J = np.polyval(I,u)

for i in range(len(I)):
    print("Coefficient of x^",i,"is:",I[i])
```

print("So the corresponding equation is: -0.8012 + 1.6754x + 7.2989x^2 - 1.9434x^3 + 6.3340x^4 + 0.0333x^5")

```
Coefficient of x^ 0 is: -0.8012820512819778
Coefficient of x^ 1 is: 1.6754079254075194
Coefficient of x^ 2 is: 7.298951048951876
Coefficient of x^ 3 is: -1.9434731934739702
Coefficient of x^ 4 is: 6.3340326340329725
Coefficient of x^ 5 is: 0.03333333333327643
So the corresponding equation is: -0.8012 + 1.6754x + 7.2989x^2 - 1.9434x^3 + 6.3340x^4 + 0.0333x^5
```

d)

```
# part d


import numpy as np

import math

import matplotlib.pyplot as plt


u = np.array([0.2,0.4,0.6,0.8,1.0,1.2,1.4,1.6,1.8,2.0])

p = np.array([1.3,2.7,4.9,8.1,12.5,18.9,27.3,37.7,50.1,64.5])


I = np.polyfit(2.7*10**8,p,5)

K = np.poly1d(I)

J = I(2.7*10**8)

print(I)
```

4. Lagrange Polynomial Interpolation

I.

a.) Finding the polynomial which goes through all the data points given as a curve and to estimate the points in between

b.) First a Lagrange Polynomial basis (L_i) is defined for each x coordinate and y coordinate and then a linear polynomial is constructed using Lagrange basis polynomials (P_i) and their corresponding values.

c.) Lagrange basis function is a function which ensures that all the interpolating polynomials goes through all the given points exactly. Lagrange basis function is equal to 1 at the given data points and is 0 at all the other data points.

Lagrange basis function assigns a weight to the function values at each data point. By taking the weighted sum of all the basis functions, the interpolating polynomial is formed so that we can ensure that the polynomial passes through all the given data points.

d.) First a set of Lagrange basis functions are constructed which are equal to 1 at their corresponding data points and equal to 0 at all the other data points. Then the interpolating polynomial is obtained as weighted sum by combining the Lagrange basis functions with the corresponding function values.

e.) By evaluating the polynomial function at the precise places, interpolating polynomials serve as an interpolating tool that enables us to estimate values at positions that lie between the known data points.

f.) Highly sensitive to data points.

Numerically unstable.

Complexity and the degree of the polynomial may increase with the number of data points given.

II.

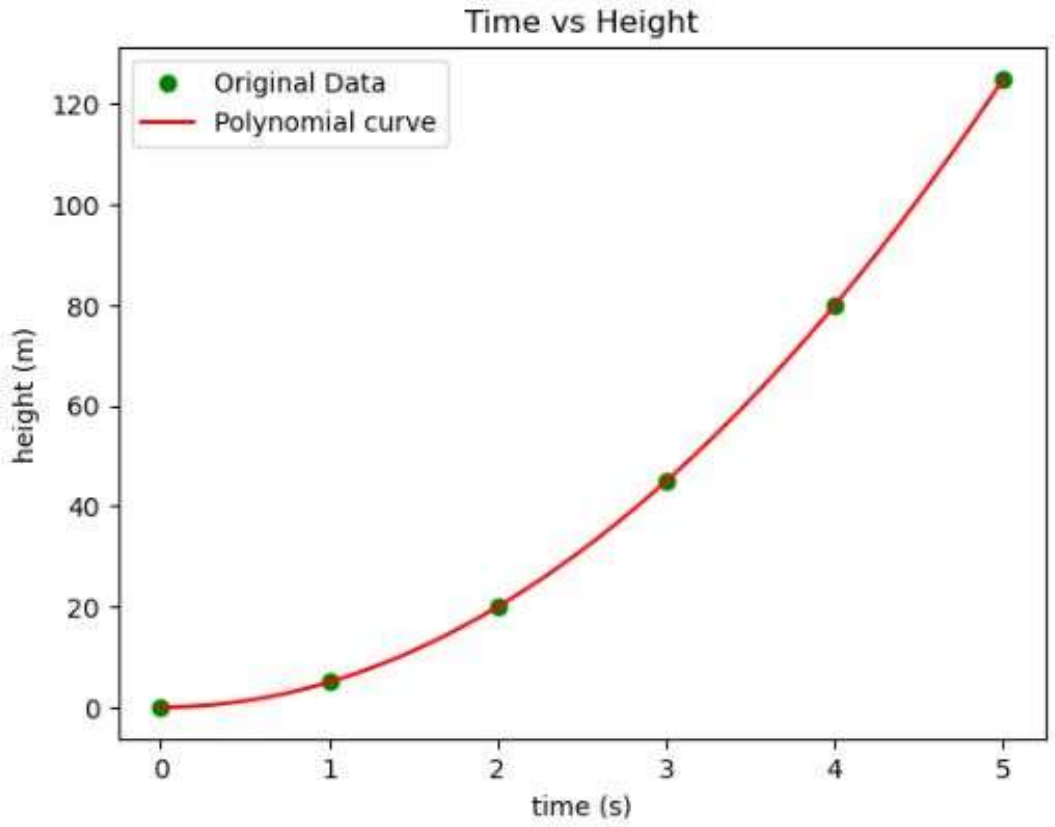
```
import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import lagrange
```

```
t = np.array([0, 1, 2, 3, 4, 5])
h = np.array([0, 5, 20, 45, 80, 125])

lag_polynomial = lagrange(t,h)
print("a)")
print("Lagrange interpolating formula:\n",lag_polynomial)
print()

# part b
print("b)")
for i in range(5):
    print("Coefficient of x^",i, "is:",lag_polynomial[i])
print()

print("c)")
h_interpolation = lag_polynomial(t)
mse = np.mean((h - h_interpolation)**2) #mean square error
print("Mean squared error:",mse)
print()

print("d)")
t_new = np.linspace(0, 5, num=100)
h_new = lag_polynomial(t_new)
plt.plot(t,h,'go', label='Original Data') #plotting original data
plt.plot(t_new,h_new,'red', label='Polynomial curve')
plt.xlabel("time (s)")
```

```
plt.ylabel("height (m)")
plt.title("Time vs Height")
plt.legend()
plt.show()

print()
print("e)")
velocity = lag_polynomial(2.5)
print("Velocity at 2.5s is:",velocity)
```

```
a)
Lagrange interpolating formula:
              5              3     2
4.441e-16 x - 1.421e-14 x + 5 x - 3.553e-15 x

b)
Coefficient of x^ 0 is: 0.0
Coefficient of x^ 1 is: -3.552713678800501e-15
Coefficient of x^ 2 is: 4.999999999999986
Coefficient of x^ 3 is: -1.4210854715202004e-14
Coefficient of x^ 4 is: 0.0

c)
Mean squared error: 2.1188803914236377e-25
```

d)



e)

```
Velocity at 2.5s is: 31.249999999999723
```

5. Linear Interpolation

I.

a.) Within the interval between two adjacent data points, the relationship between independent and dependent variables is approximately linear.

b.) Join the two consecutive data points from a straight line and then determine the value at an intermediate point on the constructed line.

c.) No

It is not reliable to do linear interpolation beyond the known data range as an interpolating function cannot be constructed without knowing the endpoints of the interval.

d.) When the distance between the target position and the known data points increases the accuracy of linear interpolation decreases as the assumption made on linearity becomes less reliable and it may arise large errors in estimated values.

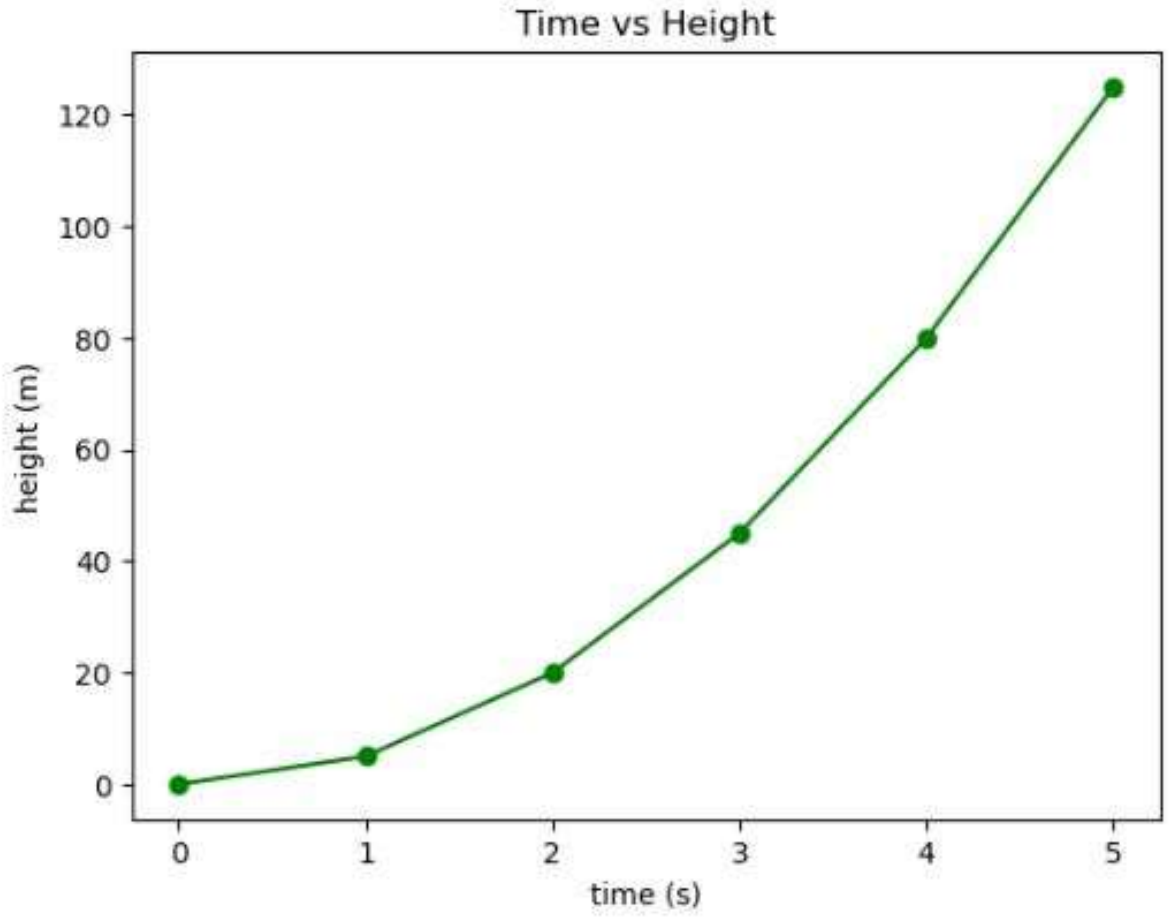
e.) In complex relationships data may be underfitted.
Sensitive to outliers
Not suitable for extrapolation beyond the data points.

f.) Linear interpolation is suitable if the data points seems to have a linear relationship or if the data points are not linear but they are very close to each other.

II.

```
#Question 5
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

t = np.array([0, 1, 2, 3, 4, 5])
h = np.array([0, 5, 20, 45, 80, 125])

lin_polynomial = interpolate.interp1d(t,h)
h_new = lin_polynomial(2.5)

plt.plot(t,h,'-go')
plt.xlabel("time (s)")
plt.ylabel("height (m)")
plt.title("Time vs Height")
plt.show()

MSE = np.mean((h-lin_polynomial(t))**2)
```

```
print("Mean squared error:",MSE)

print("Value at 2.5s is:",h_new)
```




```
Mean squared error: 0.0
Value at 2.5s is: 32.5
```