# Interpolation and Least Squares

G V V Sharma*

*Abstract*—**Through examples, this manual introduces methods for polynomial interpolation and curve fitting like Newton, Lagrange's and least squares methods. Python codes are provided for all these methods.**

## 1 Newton's Interpolation Formula

**Theorem 1.1.** Newton's interpolation formula for a function $f(x)$ is given by [1], [2]

$$f(x) = \sum_{k=0}^{\infty} \Delta^k y \, {}^uC_k \tag{0.1}$$

where

$$ {}^uC_k = \frac{u(u-1)(u-2)\cdots(u-k+1)}{k!} \tag{0.2}$$

$$\Delta^n y = \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} y_k \tag{0.3}$$

$$u = \frac{x-a}{h} \tag{0.4}$$

with $a$ as the initial point and $h$, the step-size.

**Problem 1.** For the table in Table I, obtain (0.3) for $n = 0, 1, \ldots 5$.

| | | | | | | |
|---|---|---|---|---|---|---|
| x | 0 | 0.001 | 0.002 | 0.003 | 0.004 | 0.005 |
| y | 1.121 | 1.123 | 1.1255 | 1.127 | 1.128 | 1.1285 |

TABLE I

*The author is with the Department of Electrical Engineering, IIT, Hyderabad 502285 India e-mail: gadepall@iith.ac.in. 

**Solution:** The following code generates the coefficients

$$1.121, 0.002, 0.0005, -0.0015, 0.002, -0.0025 \tag{1.1}$$

```
#Forward differences coefficients
#for Newton's Interpolation Formula
import numpy as np
from scipy.misc import comb



def diff_f(n,y):
        k = np.asarray(range(n+1))
        coeff = comb(n,k)
        return np.sum(coeff*(-1)**(n-k)*y[0:n+1])



x = np.array([0,0.001,0.002,\
0.003,0.004,0.005])
y = np.array([1.121,1.123,1.1255\
,1.127,1.128,1.1285])
n = 5

for k in range(n+1):
        print(diff_f(k,y))
```

**Problem 2.** Obtain the interpolating polynomial $P_5(u)$.

**Solution:** From (1.1) and (0.1),

$$\begin{aligned}P_5(u) &= 1.121 + u \times .002 \\ &+ \frac{u(u-1)}{2}(.0005) + \frac{u(u-1)(u-2)}{3!} \times (-.0015) \\ &+ \frac{u(u-1)(u-2)(u-3)}{4!}(.002) + \\ &\frac{u(u-1)(u-2)(u-3)(u-4)}{5!} \times (-.0025).\end{aligned} \tag{2.1}$$

**Problem 3.** Using (2.1), interpolate the value of the function at $x = 0.0045$.

**Solution:** From (0.4),

$$u = \frac{x-a}{h} = \frac{0.0045-0}{0.001} = 4.5 \tag{3.1}$$

The following code results in

$$P_5(u) = 1.128400390625 \tag{3.2}$$

```
#Forward differences
    coefficients
#for Newton's Interpolation
    Formula
import numpy as np
from scipy.misc import comb
import mpmath as mp


def diff_f(n,y):
        k = np.asarray(range(n
            +1))
        coeff = comb(n,k)
        return np.sum(coeff
            *(-1)**(n-k)*y[0:n
            +1])


x = np.array([0,0.001,0.002,
\
0.003,0.004,0.005])
y = np.array
    ([1.121,1.123,1.1255
\
,1.127,1.128,1.1285])
n = 5
u = 4.5
ans = 0
for k in range(n+1):
        ans += diff_f(k,y)*mp.
            binomial(u,k)

print(ans)
```

## 2 Lagrange's Interpolation Formula

**Theorem 2.1.** Given a set of data points

$$(x_0, y_0), \dots, (x_j, y_j), \dots, (x_k, y_k)$$

where no two $x_j$ are the same, [3],

$$f(x) = \sum_{j=0}^{k} y_j \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} \tag{3.3}$$

**Problem 4.** Using the following data [4], find by (3.3), the value of $f(x)$ at $x = 10$

| | | | | | |
|---|---|---|---|---|---|
| x | 9.3 | 9.6 | 10.2 | 10.4 | 10.8 |
| y | 11.4 | 12.8 | 14.7 | 17 | 19.8 |

TABLE II

**Solution:** The following code gives

$$f(10) = 13.1978451179 \tag{4.1}$$

```
#Forward differences
    coefficients
#for Newton's Interpolation
    Formula
import numpy as np
from scipy.interpolate import
    lagrange
from numpy.polynomial.
    polynomial import Polynomial

x = np.array([9.3,9.6,10.2,
```

```
\
10.4,10.8])
y = np.array([11.4,12.8,14.7
\
,17,19.8])

poly = lagrange(x, y)
p = Polynomial(poly).coef
print(np.polyval(p,10))
```

**Problem 5.** Using (3.3), find $x$ if $f(x) = 16$.

**Solution:** Interchange $x$ and $y$ in the previous code and evaluate f(16).

## 3 Least Squares

Use the following data for the following problems.

$$(x_i, y_i) = (0,5), (2,4), (4,1), (6,6), (8,7). \tag{5.1}$$

**Problem 6.** Obtain a matrix solution to find a line of best fit from the given data.

**Solution:** Let

$$y = mx + c \tag{6.1}$$

be the equation of the line. Using the data in (5.1) and (6.1) results in the matrix equation

$$\mathbf{Az} = \mathbf{y} \tag{6.2}$$

where

$$A = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_5 & 1 \end{pmatrix}, \mathbf{z} = \begin{pmatrix} m \\ c \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{pmatrix}, \tag{6.3}$$

The solution to (6.2) is obtained as

$$\mathbf{z} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{Ay} \tag{6.4}$$

**Problem 7.** Obtain $\mathbf{z}$ in (6.4) and sketch (6.1) using this information.

**Solution:** The following code results in

$$\mathbf{z} = \begin{pmatrix} 0.3 \\ 3.4 \end{pmatrix} \tag{7.1}$$

and plots Fig. 7.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.matrix([0,2,4,
\
6,8])
y = np.matrix([5,4,1
\
,6,7])

A = np.column_stack([x.T,np.
    ones([5,1])])

z = np.linalg.inv(A.T*A)*A.T*y.
    T
yhat=A*z
x = np.array(x)[0]
y = np.array(y)[0]
yhat = np.array(yhat.T)[0]
plt.plot(x,yhat,label='estimate
    ')
plt.plot(x,y,'o',label='data')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best', prop={'
    size':11})
plt.savefig('../figs/ls_line.
    eps')
plt.show()
```

**Problem 8.** Obtain the estimate for $y = ax^2 + bx + c$.

**Solution:** The following code results in

$$\mathbf{z} = \begin{pmatrix} 0.21428571 \\ -1.41428571 \\ 5.11428571 \end{pmatrix} \tag{8.1}$$

and plots Fig. 8.

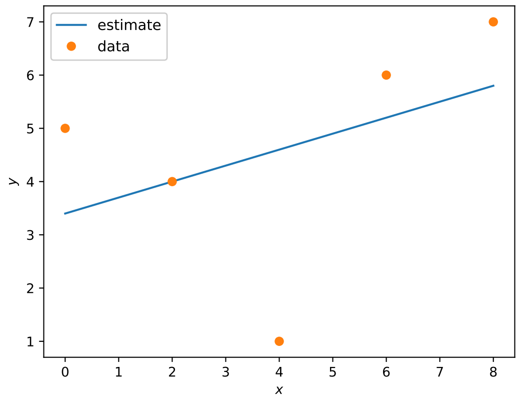


Fig. 7: Linear Estimate.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.matrix([0,2,4,
\
6,8])
x2=np.square(x)
y = np.matrix([5,4,1
\
,6,7])

A = np.column_stack([x2.T,x.T,
    np.ones([5,1])])
z = np.linalg.inv(A.T*A)*A.T*y.
    T
print(z)
yhat=A*z
x = np.array(x)[0]
y = np.array(y)[0]
yhat = np.array(yhat.T)[0]
plt.plot(x,yhat,label='estimate
    ')
plt.plot(x,y,'o',label='data')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best', prop={'
    size':11})
plt.savefig('../figs/ls_quad.
    eps')
plt.show()
```

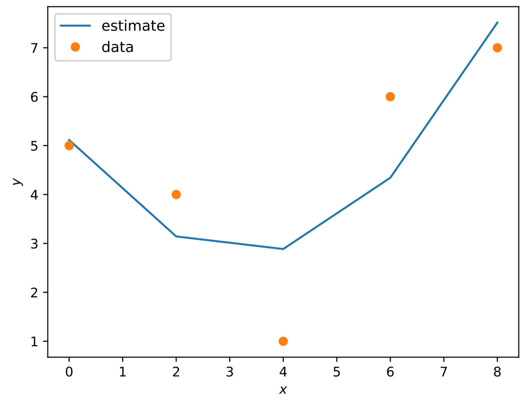


Fig. 8: Quadratic Estimate.

**Problem 9.** Write a program to find a curve of the form $y = ae^{bx}$ from the given data.

**Solution:** Taking logarithms,

$$\ln y = \ln a + bx \tag{9.1}$$

**Solution:** Note that (9.1) has the same form as (6.1). The following code results in

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.05540067 \\ 1.13099846 \end{pmatrix} \tag{9.2}$$

and plots Fig. 9.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.matrix([0,2,4,
\
6,8])
y = np.matrix([5,4,1
\
,6,7])
ylog = np.log(y)
A = np.column_stack([x.T,np.
    ones([5,1])])
z = np.linalg.inv(A.T*A)*A.T*
    ylog.T
```

```
print(np.exp(z[0]),z[1])
yloghat=A*z
yhat = np.exp(yloghat)
x = np.array(x)[0]
y = np.array(y)[0]
yhat = np.array(yhat.T)[0]
plt.plot(x,yhat,label='estimate
    ')
plt.plot(x,y,'o',label='data')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best', prop={'
    size':11})
#plt.savefig('../figs/ls_exp.
    eps')
plt.show()
```

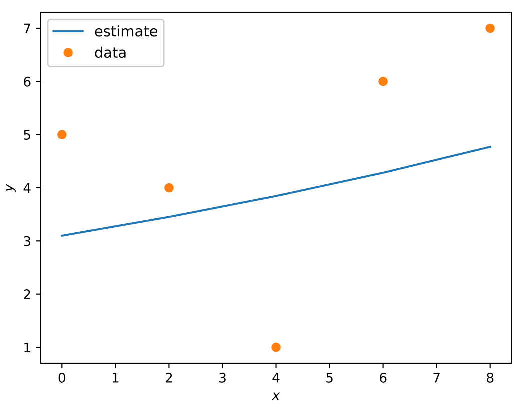


Fig. 9: Exponential Estimate.

**Problem 10.** Write a program to find a curve of the form $y = ax^b$ from the given data.

**Solution:** Taking logarithms,

$$\ln y = \ln a + b \ln x \tag{10.1}$$

**Solution:** Note that (10.1) has the same form as (6.1). The following code results in

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.00450417 \\ 1.34195488 \end{pmatrix} \tag{10.2}$$

and plots Fig. 10.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.matrix([0.5,2,4,
\
6,8])
y = np.matrix([5,4,1
\
,6,7])
ylog = np.log(y)
xlog = np.log(x)
A = np.column_stack([xlog.T,np.
    ones([5,1])])
z = np.linalg.inv(A.T*A)*A.T*
    ylog.T
print(np.exp(z[0]),z[1])
yloghat=A*z
yhat = np.exp(yloghat)
x = np.array(x)[0]
y = np.array(y)[0]
yhat = np.array(yhat.T)[0]
plt.plot(x,yhat,label='estimate
    ')
plt.plot(x,y,'o',label='data')
plt.xlabel('$x$')
plt.ylabel('$y$')
#plt.legend(loc='best', prop={'
    size':11})
plt.savefig('../figs/ls_pow.eps
    ')
plt.show()
```

## References


[1] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Finite_difference

[2] NPTEL. [Online]. Available: http://www.nptel.ac.in/courses/122104018/node109.html

[3] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Lagrange_polynomial

[4] NPTEL. [Online]. Available: http://www.nptel.ac.in/courses/122104018/node113.html

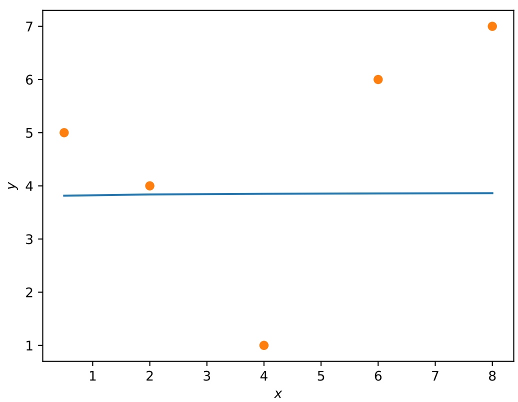


Fig. 10: Estimate.