In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
# yi = mxi+ b +ei
# design matrix
# A = [np.ones(len(t)), x] B=[b m]
t = np.array([0, 1, 2, 3, 4])
y = np.array([0.0434, 1.0343, -0.2588, 3.68622, 4.3188])
y_errors = np.array([0.1, 0.1, np.nan, np.nan, np.nan])

A = np.vstack([np.ones(len(t)), t]).T  # Design matrix
```

In [8]:
```
m, b = np.linalg.lstsq(A, y, rcond=None)[0]
```

In [10]:
```
residuals = y[2:] - (m * t[2:] + b)
variance_estimate = np.mean(residuals**2)

# Output initial m, b, and estimated error variance
m, b, variance_estimate
```

Out[10]:
```
(-0.47575999999999985, 1.1202720000000002, 14.051555543770663)
```

In [3]:
```
# try with bayesian
# try another method
t = np.array([0, 1, 2, 3, 4])
y = np.array([0.0434, 1.0343, -0.2588, 3.68622, 4.3188])

# Creating the design matrix A
ones = np.ones(len(t))  # Match the length of t
A = np.column_stack((ones, t))  # This creates your design matrix for linear regressi


# Performing the linear regression calculation, best estimate of B
B_estimated = np.linalg.inv(A.T @ A) @ A.T @ y

print("Estimated parameters (mu, b):", B_estimated)
```

```
Estimated parameters (mu, b): [-0.47576   1.120272]
```

In [7]:
```
residual = A@B_estimated -y
sigma_sqrt= np.dot(residual, residual.T)
```

In [9]:
```
sigma_sqrt = sigma_sqrt/6 #5+1=N+1=6
sigma_sqrt
```

Out[9]:
```
0.14601340125777781
```

In [11]:
```
sigma = np.sqrt (sigma_sqrt)
sigma
```

Out[11]:
```
0.3821169994357459
```

In [ ]: