ANSWER TO THE QUSTION NO. 6

In [1]:
```
#import libraries
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
t = np.arange(0, 15, 0.01)
t
```

Out[2]:
```
array([0.000e+00, 1.000e-02, 2.000e-02, ..., 1.497e+01, 1.498e+01,
       1.499e+01])
```

In [3]:
```
len(t)
```

Out[3]:
```
1500
```

In [4]:
```
# Declare Final time T
T= 15
Tsq = np.power(T,2)
Tcb = np.power(T,3)
```

In [5]:
```
#initialized A
A = np.array([[1, 0, 0, 0,  0, 0, 0, 0],
              [0, 1, 0, 0,  0, 0, 0, 0],
              [0, 0, 0, 0,  1, 0, 0, 0],
              [0, 0, 0, 0,  0, 1, 0, 0],
              [1, T, Tsq, Tcb,  0, 0, 0, 0],
              [0, 1, 2*T, 3*Tsq,  0, 0, 0, 0],
              [0, 0, 0, 0,  1, T, Tsq, Tcb,],
              [0, 0, 0, 0,  0, 1, 2*T, 3*Tsq]
              ]
            )
A
```

Out[5]:
```
array([[   1,    0,    0,    0,    0,    0,    0,    0],
       [   0,    1,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    1,    0,    0,    0],
       [   0,    0,    0,    0,    0,    1,    0,    0],
       [   1,   15,  225, 3375,    0,    0,    0,    0],
       [   0,    1,   30,  675,    0,    0,    0,    0],
       [   0,    0,    0,    0,    1,   15,  225, 3375],
       [   0,    0,    0,    0,    0,    1,   30,  675]])
```

In [6]:
```
# A pesudo inverse
Ainv = np.linalg.pinv(A)
Ainv
```

Out[6]:
```
array([[ 1.00000000e+00,  6.58070820e-13,  0.00000000e+00,
         0.00000000e+00, -8.64325972e-16,  3.91841507e-15,
         0.00000000e+00,  0.00000000e+00],
       [ 4.16333634e-17,  1.00000000e+00,  0.00000000e+00,
         0.00000000e+00, -1.00180281e-16,  1.21430643e-17,
         0.00000000e+00,  0.00000000e+00],
       [-1.33333333e-02, -1.33333333e-01,  0.00000000e+00,
         0.00000000e+00,  1.33333333e-02, -6.66666667e-02,
         0.00000000e+00,  0.00000000e+00],
       [ 5.92592593e-04,  4.44444444e-03,  0.00000000e+00,
         0.00000000e+00, -5.92592593e-04,  4.44444444e-03,
         0.00000000e+00,  0.00000000e+00],
       [ 4.38264820e-17, -1.56199568e-17,  1.00000000e+00,
         4.13738488e-13, -4.38264820e-17,  2.19459069e-16,
         3.73236598e-16, -9.14524532e-16],
       [-3.59628264e-18, -8.65405541e-17, -1.66533454e-16,
         1.00000000e+00,  3.59628264e-18, -1.80076414e-17,
        -3.85975973e-17,  7.75421394e-16],
       [-1.03997980e-19,  1.18523646e-17, -1.33333333e-02,
        -1.33333333e-01,  1.03997980e-19, -5.20842611e-19,
         1.33333333e-02, -6.66666667e-02],
       [ 9.93178042e-21, -4.00814858e-19,  5.92592593e-04,
         4.44444444e-03, -9.93178042e-21,  4.97355363e-20,
        -5.92592593e-04,  4.44444444e-03]])
```

In [7]:
```python
#initialized b
b =  np.array([[0],#x1(0)
               [0],#1
               [0],#x3(0)
               [-0.5],#x2(0)
               [5],#x1(T)
               [0],#1
               [5],#x3(T)
               [-0.5] #x2(T)
              ]
            )

b
```

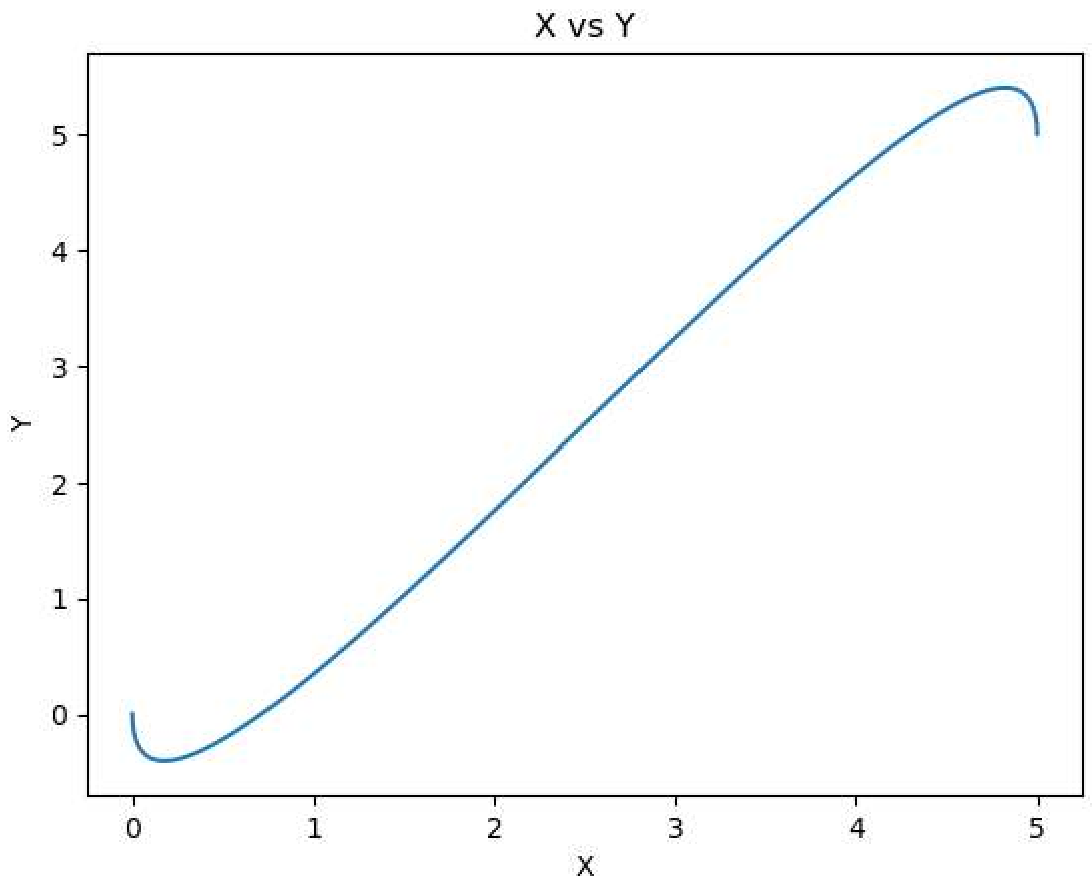
Out[7]:
```
array([[ 0. ],
       [ 0. ],
       [ 0. ],
       [-0.5],
       [ 5. ],
       [ 0. ],
       [ 5. ],
       [-0.5]])
```

In [8]:
```python
#matrix multiplication x = Ainv * b
x= np.matmul(Ainv, b)
x
```

Out[8]:
```
array([[-4.32162986e-15],
       [-5.00901404e-16],
       [ 6.66666667e-02],
       [-2.96296296e-03],
       [-2.04764931e-13],
       [-5.00000000e-01],
       [ 1.66666667e-01],
       [-7.40740741e-03]])
```

In [9]:
```python
a11 = x[0]
a11
```

```
Out[9]: array([-4.32162986e-15])

In [10]: a12 = x[1]
         a12

Out[10]: array([-5.00901404e-16])

In [11]: a13 = x[2]
         a13

Out[11]: array([0.06666667])

In [12]: a14 = x[3]
         a14

Out[12]: array([-0.00296296])

In [13]: a21 = x[4]
         a21

Out[13]: array([-2.04764931e-13])

In [14]: a22 = x[5]
         a22

Out[14]: array([-0.5])

In [15]: a23 = x[6]
         a23

Out[15]: array([0.16666667])

In [16]: a24 = x[7]
         a24

Out[16]: array([-0.00740741])

In [17]: for i in t:
           X = a11 + (a12* t) + (a13 * np.power(t,2)) + (a14 * np.power(t,3))
           Y = a21 + (a22* t) + (a23 * np.power(t,2)) + (a24 * np.power(t,3) )

         X, Y

Out[17]: (array([-4.32162986e-15,  6.66370370e-06,  2.66429630e-05, ...,
                  4.99994008e+00,  4.99997336e+00,  4.99999334e+00]),
          array([-2.04764931e-13, -4.98334074e-03, -9.93339259e-03, ...,
                  5.01485020e+00,  5.00993339e+00,  5.00498334e+00]))

In [18]: plt.plot(X,Y)
         plt.title('X vs Y')
         plt.xlabel('X')
         plt.ylabel('Y')

Out[18]: Text(0, 0.5, 'Y')
```



In [19]:
```
X_new = a11 + a12 * t + a13 * t**2 + a14 * t**3
Y_new = a21 + a22 * t + a23 * t**2 + a24 * t**3
```

In [20]:
```
Xdd = np.gradient(np.gradient(X_new, t), t)
Ydd = np.gradient(np.gradient(Y_new, t), t)
```

In [21]:
```
theta = np.arctan2(np.gradient(Y_new, t), np.gradient(X_new, t))
V = np.sqrt(np.gradient(X_new, t)**2 + np.gradient(Y_new, t)**2)
a = np.cos(theta) * Xdd + np.sin(theta) * Ydd

omega = (-np.sin(theta) * Xdd + np.cos(theta) * Ydd) / V
```

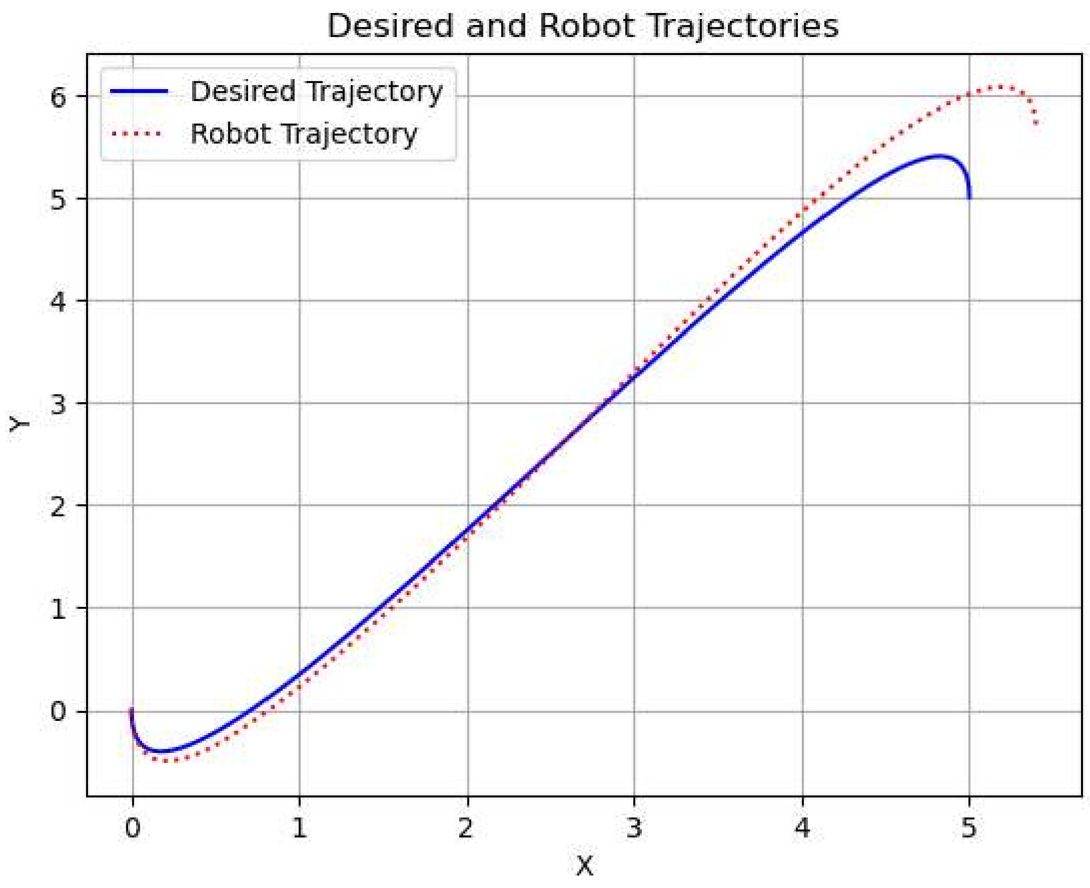
In [22]:
```
noise_std_v = 0.01
noise_std_theta = 0.001
noise_v = np.random.normal(0, noise_std_v, len(t))
noise_theta = np.random.normal(0, noise_std_theta, len(t))
```

In [23]:
```
# initialize
# x_final = X_new[0]
# y_final = Y_new[0]
# theta_final = theta[0]
# V_final = V[0]
x_final = X_new[0]
y_final = Y_new[0]
theta_final = theta[0]
V_final = V[0]
```

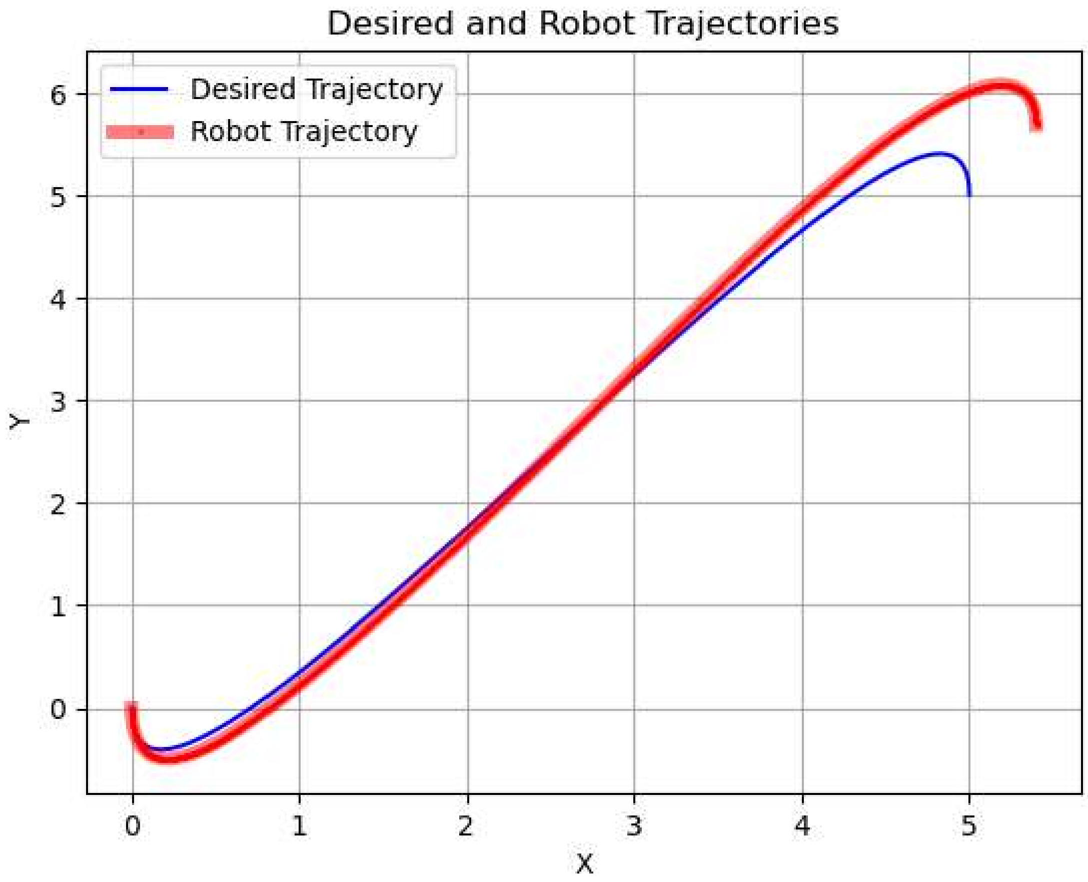
In [24]:
```
x_states = [x_final]
y_states = [y_final]
```

In [25]:
```
# for i in range(1, len(t)):
#     x_final += V_final * np.cos(theta_final) * (t[i] - t[i-1])
#     y_final += V_final * np.sin(theta_final) * (t[i] - t[i-1])
#     theta_final += omega[i] * (t[i] - t[i-1])
#     V_final += a[i] * (t[i] - t[i-1])

#     x_states.append(x_final)
#     y_states.append(y_final)
for i in range(1, len(t)):
    x_final += V_final * np.cos(theta_final) * (t[i] - t[i-1])
    y_final += V_final * np.sin(theta_final) * (t[i] - t[i-1])

    # Add noise to theta and V
    theta_final += omega[i] * (t[i] - t[i-1]) + np.random.normal(0, noise_std_thet
    V_final += a[i] * (t[i] - t[i-1]) + np.random.normal(0, noise_std_v)

    x_states.append(x_final)
    y_states.append(y_final)
```

In [26]:
```
plt.figure()
plt.plot(X, Y, label='Desired Trajectory', color='blue')
plt.plot(x_states, y_states, label='Robot Trajectory', linestyle='dotted', color='
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Desired and Robot Trajectories')
plt.grid(True)
plt.show()
```



In [27]:
```
plt.figure()
plt.plot(X, Y, label='Desired Trajectory', color='blue')
plt.plot(x_states, y_states, label='Robot Trajectory', linestyle='-', linewidth=5,
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Desired and Robot Trajectories')
plt.grid(True)
plt.show()
```



```
In [ ]:
```