1. Import the required Python modules: NumPy, Matplotlib, and SciPy's optimization library.
2. Define input data as arrays t (time) and u (voltage across a capacitor)
3. Define the model function that will fit the data. The decay_linear_func function combines an exponential decay term and a linear trend, and takes as input the time variable t and four parameters A, B, C, and D. The function returns the output variable UC that represents the voltage across a capacitor in an RLC circuit.

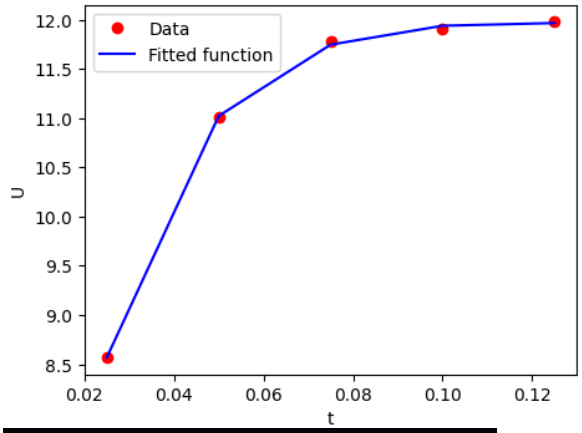
a.
```
# Define the model function to fit the data
def decay_linear_func(t, A, B, C, D):
    # Optional: Print the parameter values for debugging
    # print(A, B)
    return A * np.exp(-t / B) + C * t + D      You, 18 m
```

4. Fit the model function to the data using curve_fit from SciPy's optimization library. The curve_fit function optimizes the values of the parameters to minimize the difference between the model function and the data points. The initial guess for the parameters p0 is set to (12, 0.05, 0, 0).

a. The p0 parameter is set to (12, 0.05, 0, 0), which specifies an initial guess of A = 12, B = 0.05, C = 0, and D = 0. These values are chosen based on a visual inspection of the data.

b.
```
# Fit the model function to the data using curve_fit
x, y = opt.curve_fit(decay_linear_func, t, u, p0=(12, 0.05, 0, 0))
```

5. Send the acquired data back to the decay_linear_func to calculate the best fit for Uc(t)
6. Plots the given points and the best fit.
7. We can see that the fitted function UC(t) is of the same form as the solution to the differential equation, with A = x[0] and RC = x[1]. Therefore, to calculate the value of RC, we can use the value of x[1] obtained from curve_fit()
8. Prints RC value.



After comparing the data points and the fitted function it is accurate.

```
RC =  0.021241635874803978
```
RC value