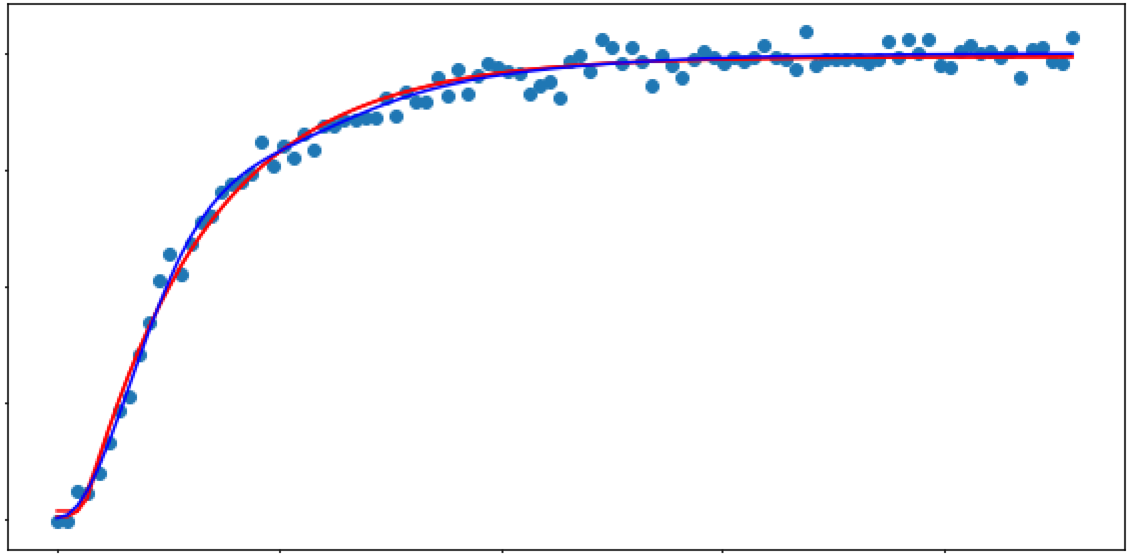
FIRST ORDER PLUS DEAD TIME REGRESSION

Optimizing parameter choices



UNIVERSITY OF WATERLOO

LEARNING OUTCOMES

- Understand the basic principles of fitting models to data via least squares estimation
- Fit FOPDT models to process data using least squares optimization with Scipy

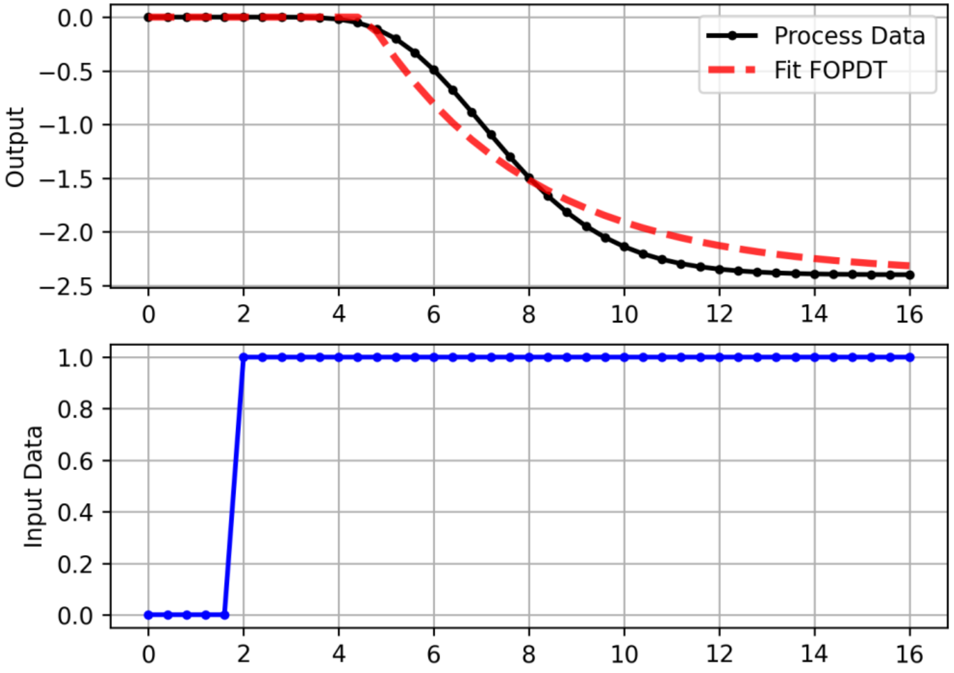
RESOURCES

- APMonitor course
    - https://apmonitor.com/pdc/index.php/Main/FirstOrderOptimization

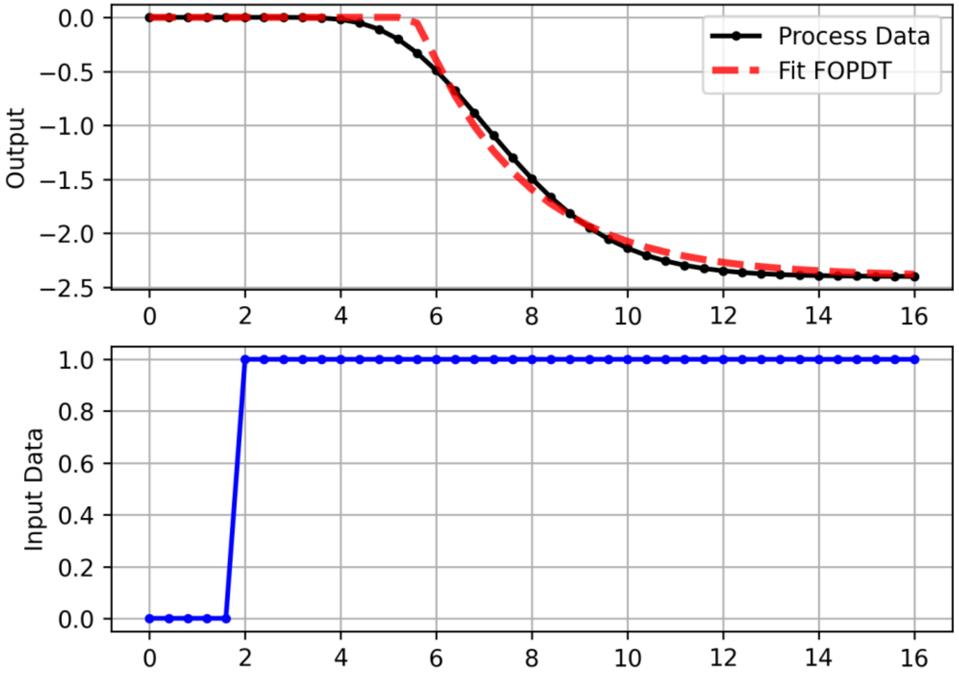
LAST TIME: FOPDT GRAPHICAL FITTING

- Limited to process data generated from a step test
- Doesn't necessarily find the best possible parameters

$$\tau_p \frac{dy'(t)}{dt} = -y'(t) + K_p u'\left(t - \theta_p\right)$$



trial and error



How can we find the "best" parameters for any data without guessing?

LEAST SQUARES REGRESSION

- Data of input/output pairs

$$\{(u_i, y_{\text{data},i}) : i = 1, 2, \ldots, n\}$$

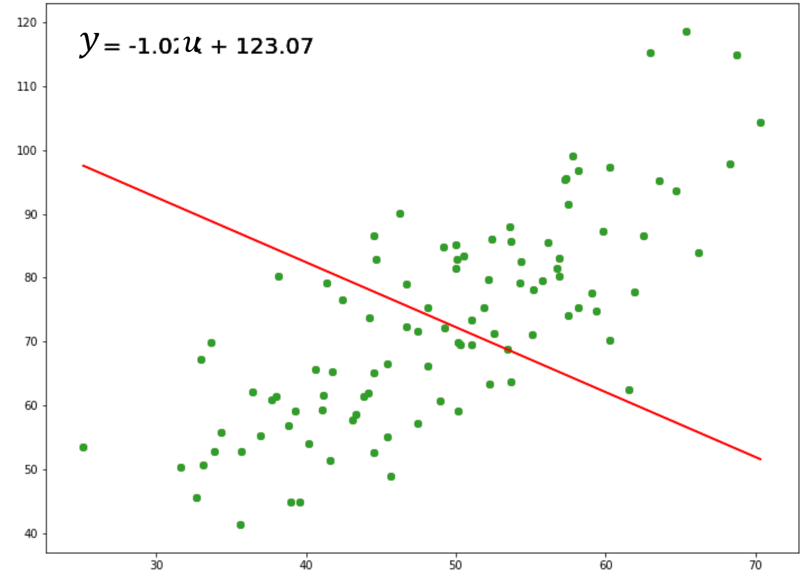
- Model f w/ parameters p that predicts output

$$y_{\text{pred},i} = f(u_i; x)$$

- Minimize the sum of squared errors (SSE) by adjusting p

$$\min_x \sum_i \left(y_{\text{data},i} - y_{\text{pred},i}\right)^2 = \sum_i \left(y_{\text{data},i} - f(u_i; x)\right)^2$$

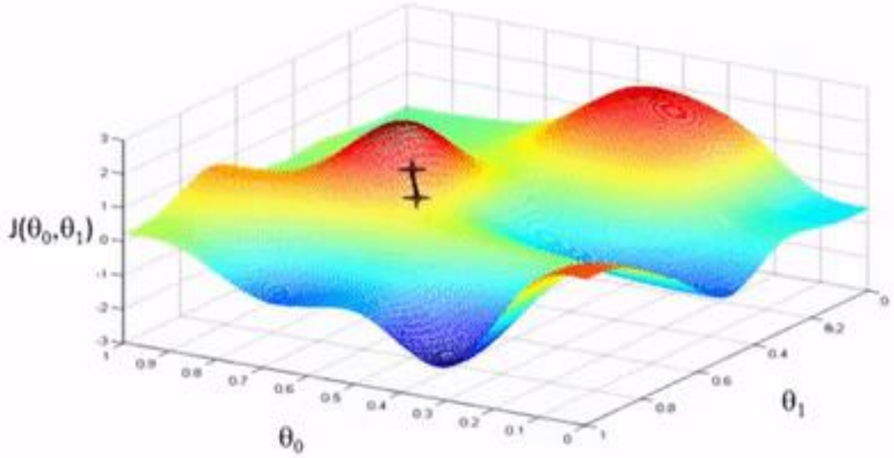
- Use optimization to effectively minimize



OPTIMIZATION BASICS

- Mathematics of decision making
- *Data:* Fixed values we cannot change
- *Variables:*
  - The values (i.e., choices) we wish to adjust
  - Often have lower/upper limits called bounds
- *Objective:*
  - A mathematical function that we can minimize
  - Determines what makes certain choices better
- Constraints: Later in the semester

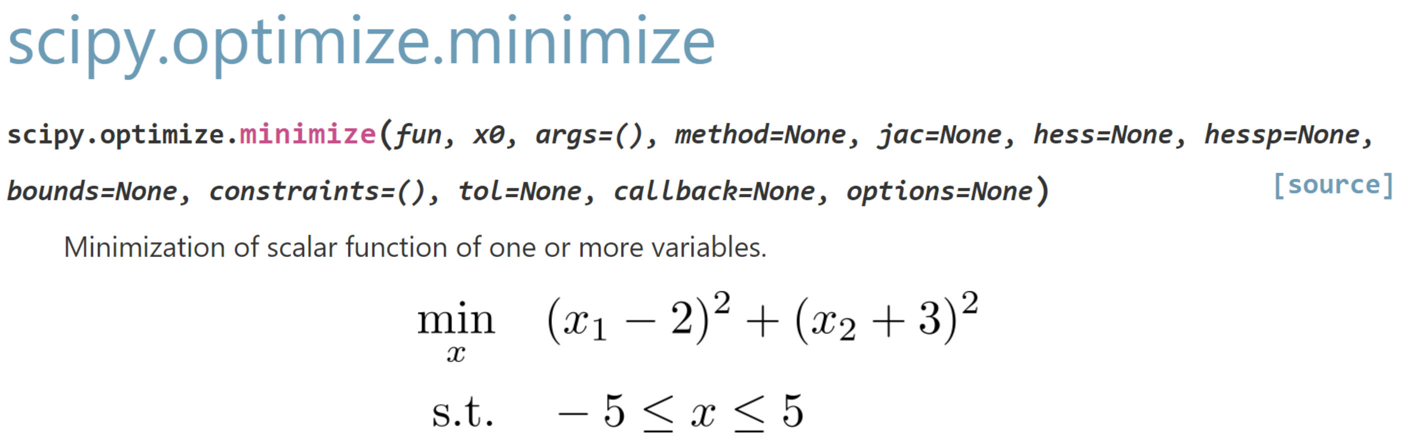
$$\begin{aligned} \min_{x} \quad & f(x;p) \\ \text{s.t.} \quad & \underline{x} \le x \le \overline{x} \end{aligned}$$



Andrew Ng

OPTIMIZING W/ SCIPY

- Define an objective function *fun*
  - Takes vector variable *x* as input
  - Returns scalar objective value
- Define initial guess *x0*
- Provide variable bounds if possible



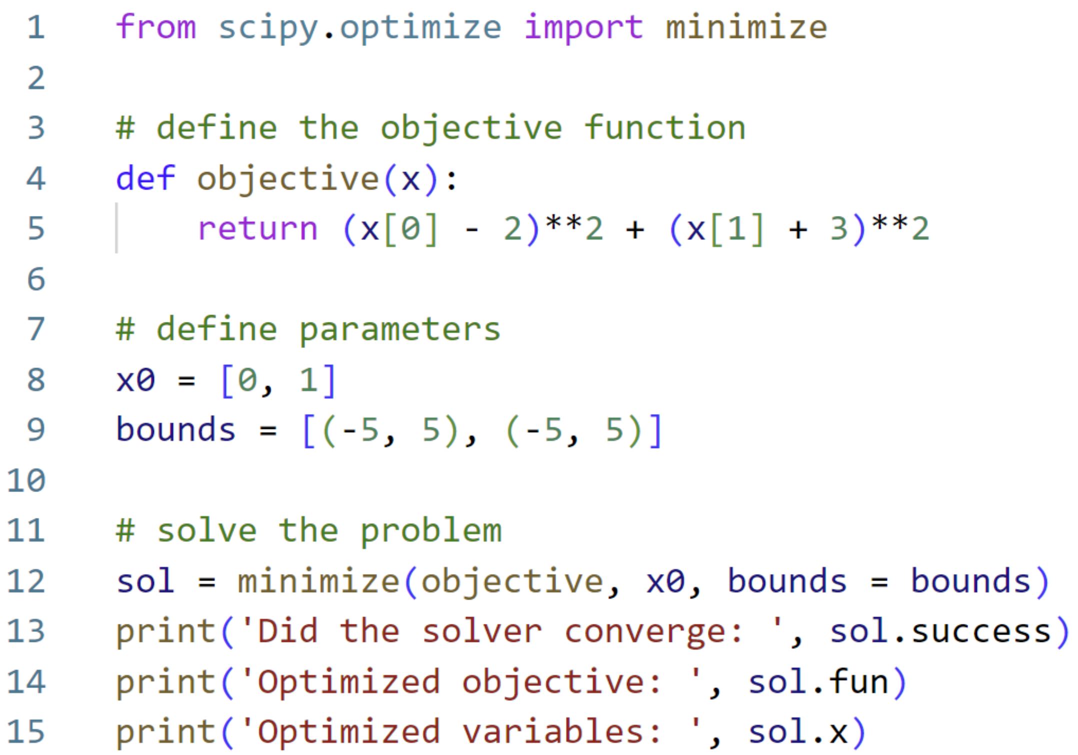
scipy.optimize.minimize

scipy.optimize.minimize(*fun, x0, args=(), method=None, jac=None, hess=None, hessp=None, bounds=None, constraints=(), tol=None, callback=None, options=None*) [source]

Minimization of scalar function of one or more variables.

$$\min_{x} \quad (x_1 - 2)^2 + (x_2 + 3)^2$$

$$\text{s.t.} \quad -5 \leq x \leq 5$$



```
from scipy.optimize import minimize

# define the objective function
def objective(x):
    return (x[0] - 2)**2 + (x[1] + 3)**2

# define parameters
x0 = [0, 1]
bounds = [(-5, 5), (-5, 5)]

# solve the problem
sol = minimize(objective, x0, bounds = bounds)
print('Did the solver converge: ', sol.success)
print('Optimized objective: ', sol.fun)
print('Optimized variables: ', sol.x)
```

UNIVERSITY OF WATERLOO

FOPDT REGRESSION

- Dynamic process data

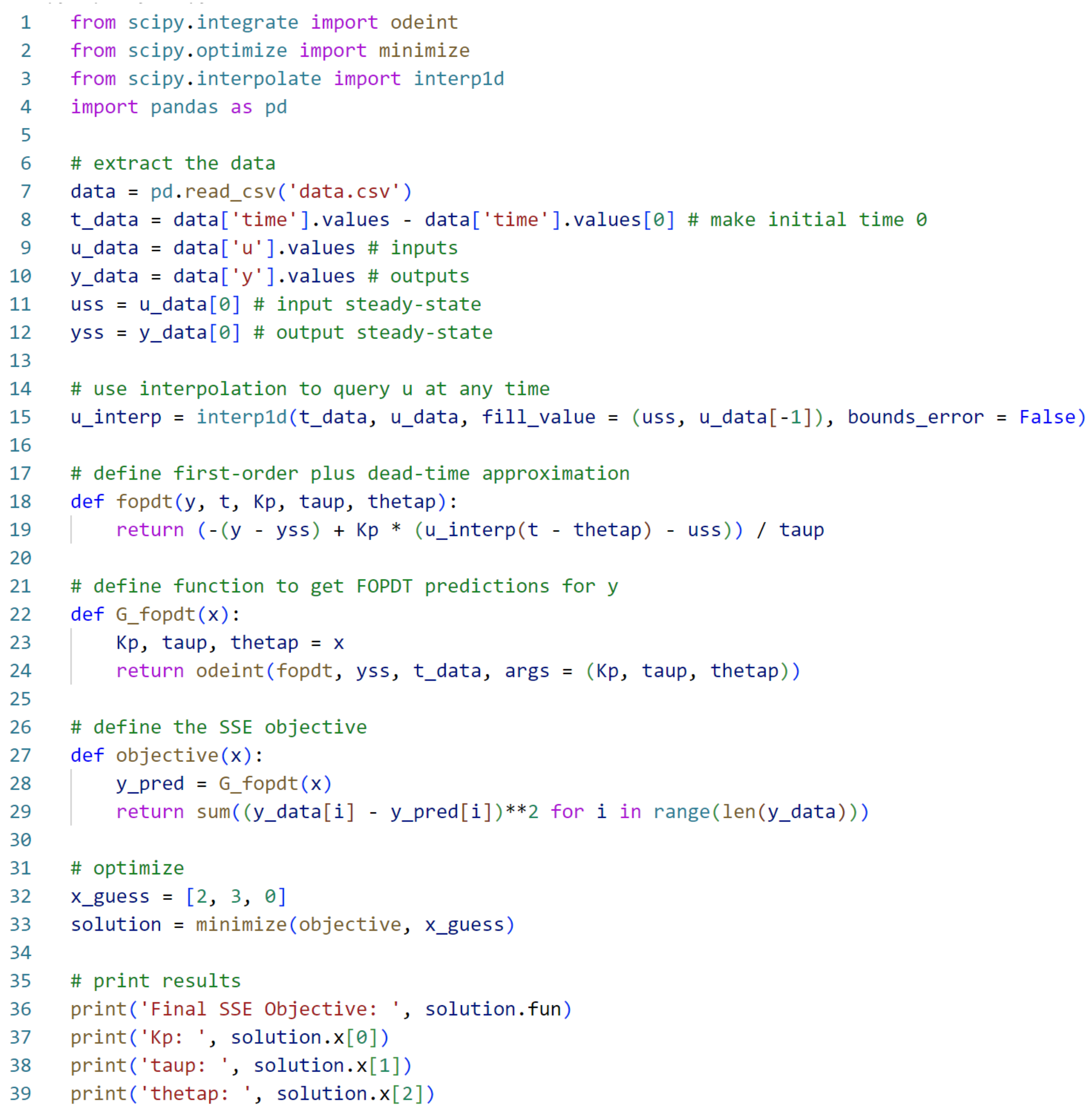
$$\{(t_i, u_i, y_{\text{data},i}) : i = 1, 2, \ldots, n\}$$

- Create a function to integrate FOPDT model that uses the data
  - Inputs are parameters, outputs are the predictions at the data time points

$$y_{\text{pred}} = G_{\text{fopdt}}(K_p, \tau_p, \theta_p)$$

- Minimize the SSE between the predictions and the data

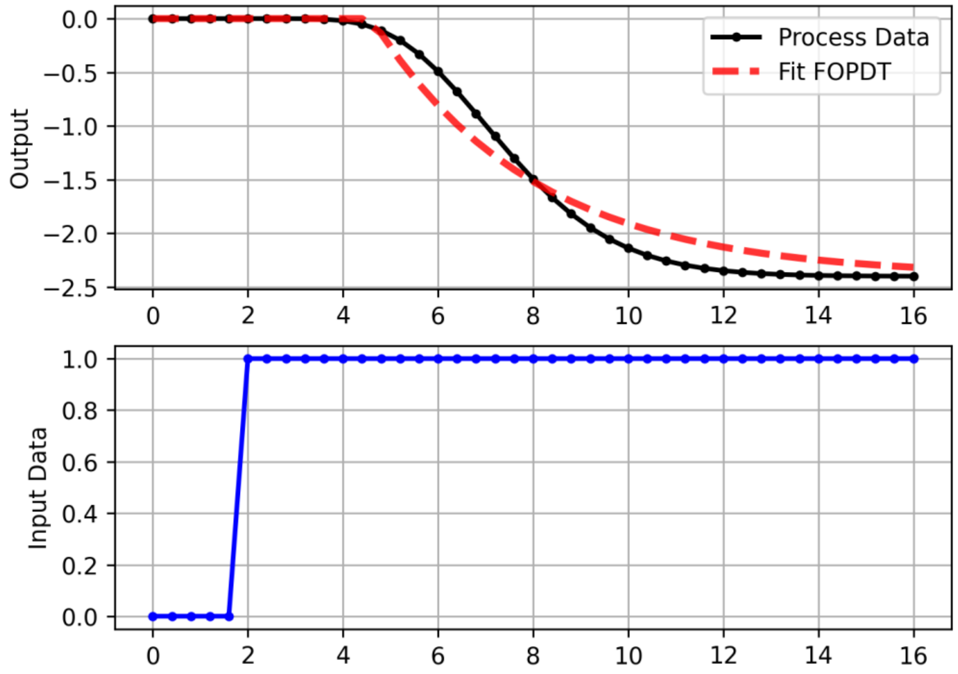
$$\min_{K_p, \tau_p, \theta_p} \sum_i \left(y_{\text{data},i} - G_{\text{fopdt},i}(K_p, \tau_p, \theta_p)\right)^2$$



```python
from scipy.integrate import odeint
from scipy.optimize import minimize
from scipy.interpolate import interp1d
import pandas as pd

# extract the data
data = pd.read_csv('data.csv')
t_data = data['time'].values - data['time'].values[0] # make initial time 0
u_data = data['u'].values # inputs
y_data = data['y'].values # outputs
uss = u_data[0] # input steady-state
yss = y_data[0] # output steady-state

# use interpolation to query u at any time
u_interp = interp1d(t_data, u_data, fill_value = (uss, u_data[-1]), bounds_error = False)

# define first-order plus dead-time approximation
def fopdt(y, t, Kp, taup, thetap):
    return (-(y - yss) + Kp * (u_interp(t - thetap) - uss)) / taup

# define function to get FOPDT predictions for y
def G_fopdt(x):
    Kp, taup, thetap = x
    return odeint(fopdt, yss, t_data, args = (Kp, taup, thetap))

# define the SSE objective
def objective(x):
    y_pred = G_fopdt(x)
    return sum((y_data[i] - y_pred[i])**2 for i in range(len(y_data)))

# optimize
x_guess = [2, 3, 0]
solution = minimize(objective, x_guess)

# print results
print('Final SSE Objective: ', solution.fun)
print('Kp: ', solution.x[0])
print('taup: ', solution.x[1])
print('thetap: ', solution.x[2])
```

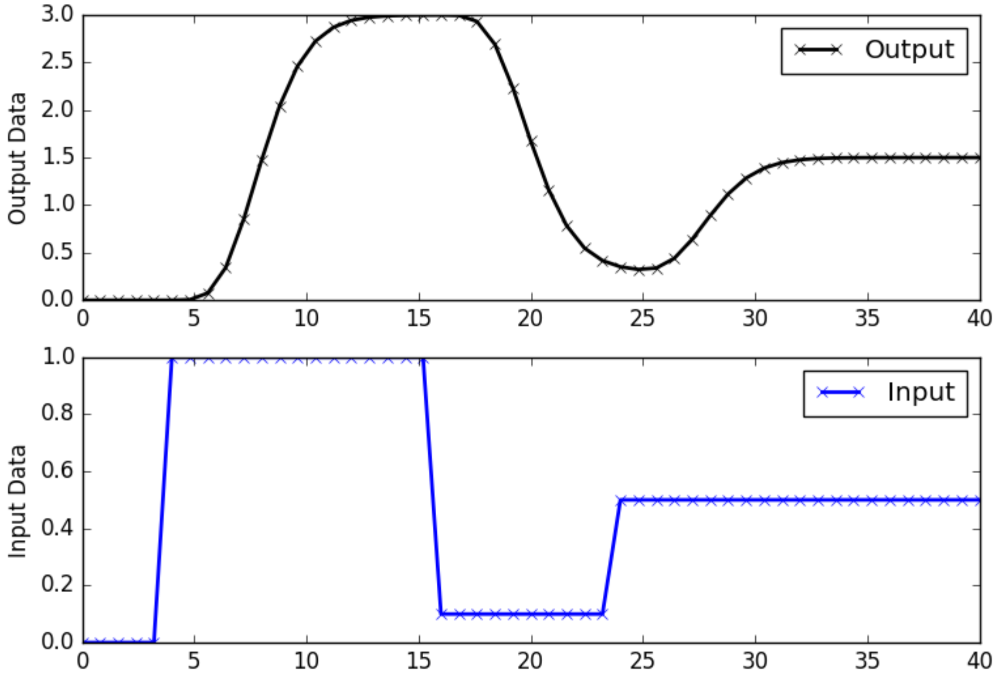
EXAMPLE: IMPROVE GRAPHICAL FIT

- Recall FOPDT fit we achieved with parameters from graphical method
- How does the optimized fit compare?



EXERCISE

- Use Scipy to fit an FOPDT model to the data
- Use the starter script “lecture6_starter.py”



BEFORE NEXT TIME

- Quiz 4: Due at 11:59pm
- Assignment 2: Due Monday (same time as Test 1)
- Study for Test 1
  - https://apmonitor.com/pdc/index.php/Main/ExamModeling