# Alex Iliadis - Week 7 Solutions

April 29, 2024

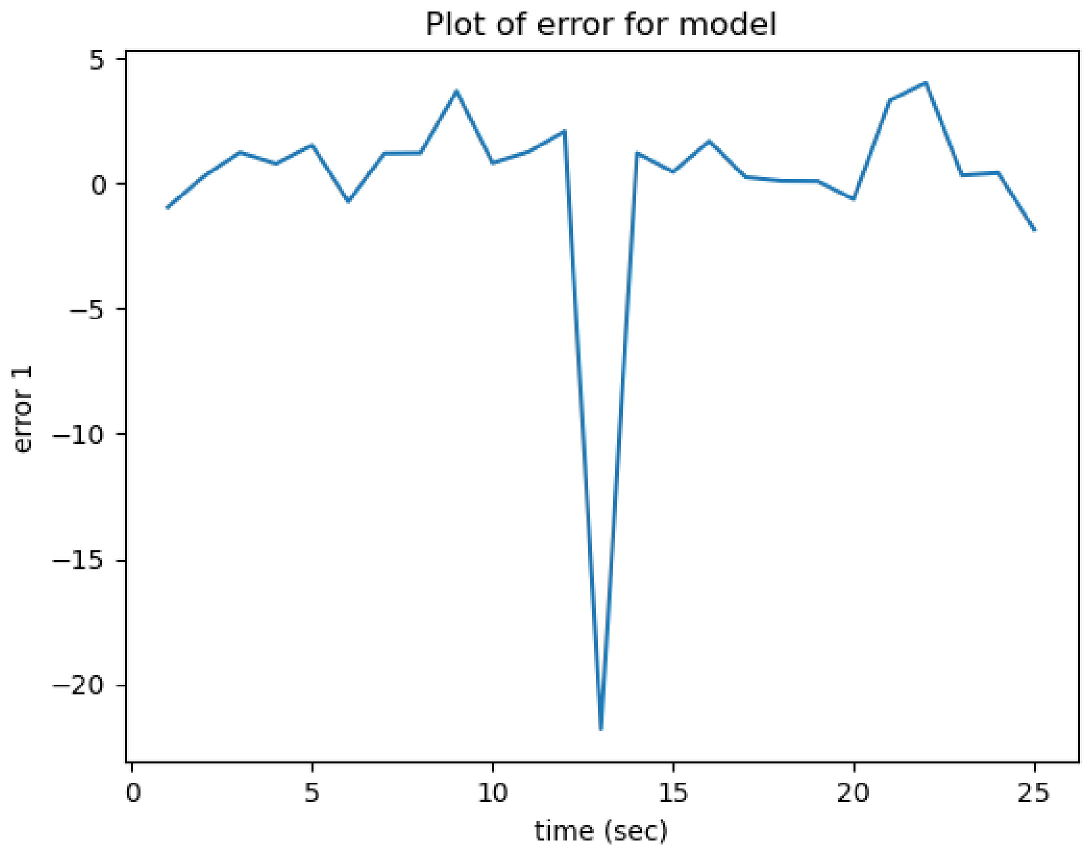
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

[2]:
```
t = np.arange(1, 26)
y = np.array([12.1739, 15.8046, 19.1459, 21.0933, 24.2243, 24.3662, 28.6772, 31.
 ↪0866,
35.9575, 35.4795, 38.3172, 41.5313, 20.0508, 45.4348, 47.0831, 50.7045,
51.6634, 53.9006, 56.2827, 57.9554, 64.2971, 67.3964, 66.0845, 68.5797,
68.7091])
```

[3]:
```
def fit_straight_line(t, y):
    X1 = np.vstack((np.ones(len(t)), np.array(t))).T
    beta = np.linalg.lstsq(X1, y, rcond=None)[0]
    u = np.arange(0, 26.01, 0.01)
    v2 = np.vstack((np.ones(len(u)), np.array(u))).T @ beta
    res = y - X1 @ beta
    resid = np.linalg.norm(res, 2)**2
    return resid, res, beta, v2
```

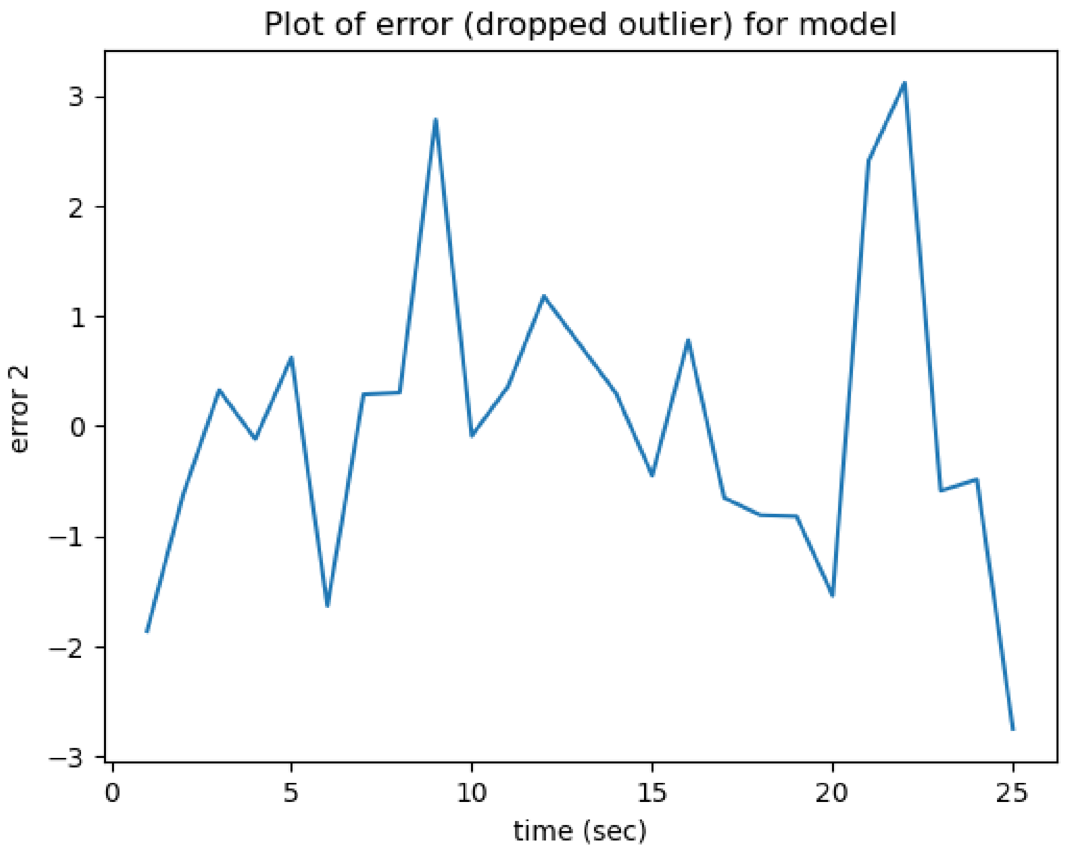
[4]:
```
resid, res1, beta, _ = fit_straight_line(t, y)

plt.plot(t, res1)
plt.xlabel('time (sec)')
plt.ylabel('error 1')
plt.title('Plot of error for model')
```

[4]:
```
Text(0.5, 1.0, 'Plot of error for model')
```



```
[5]: idx_large_error, = np.where(res1 == min(res1))
     y_dropped = np.delete(y, idx_large_error)
     t_dropped = np.delete(t, idx_large_error)
     resid, res2, beta, v1 = fit_straight_line(t_dropped, y_dropped)

     plt.plot(t_dropped, res2)
     plt.xlabel('time (sec)')
     plt.ylabel('error 2')
     plt.title('Plot of error (dropped outlier) for model')
```

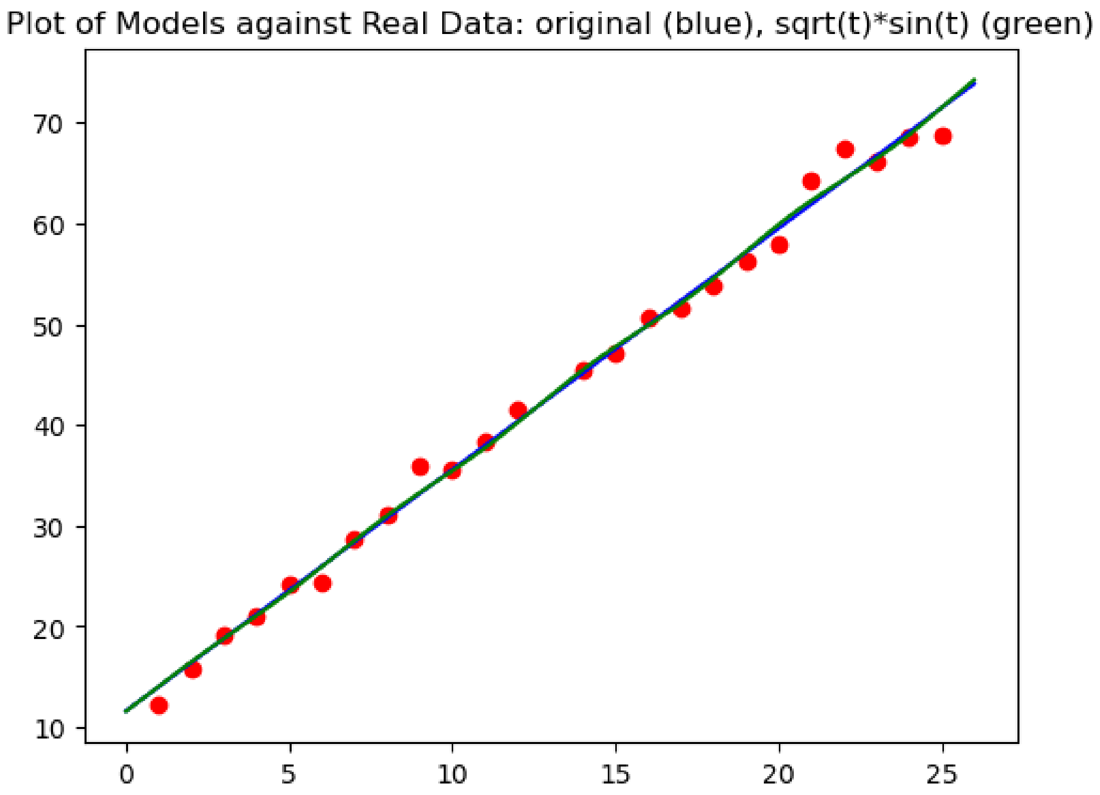
```
[5]: Text(0.5, 1.0, 'Plot of error (dropped outlier) for model')
```



The residuals here seem to be oscillating above and below 0.

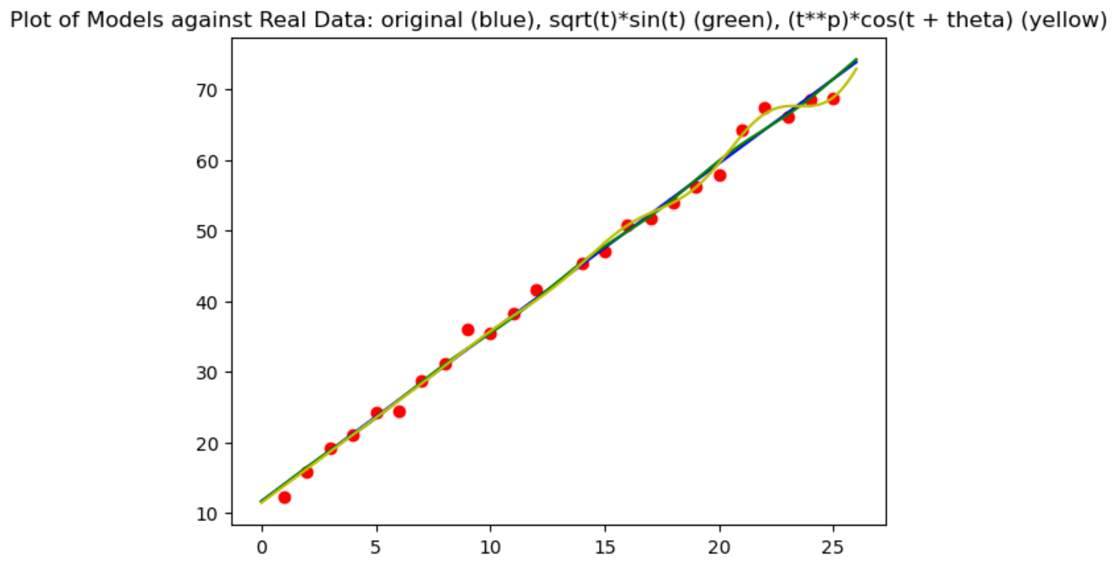
```
[6]: def fit_straight_line_w_sin(t, y):
         X1 = np.vstack((np.ones(len(t)), np.array(t), np.sqrt(t)*np.sin(t))).T
         beta = np.linalg.lstsq(X1, y, rcond=None)[0]
         u = np.arange(0, 26.01, 0.01)
         v2 = np.vstack((np.ones(len(u)), np.array(u), np.sqrt(u)*np.sin(u))).T @␣
      ↪beta
         res = y - X1 @ beta
         resid = np.linalg.norm(res, 2)**2
         return resid, res, beta, v2

     resid, res3, beta, v2 = fit_straight_line_w_sin(t_dropped, y_dropped)

     u = np.arange(0, 26.01, 0.01)
     plt.figure()
     plt.plot(t_dropped, y_dropped, 'ro')
     plt.plot(u, v1, 'b-', u, v2, 'g-')
     plt.title('Plot of Models against Real Data: original (blue), sqrt(t)*sin(t)␣
      ↪(green)')
```

```
plt.show()
```



[7]:
```
def fit_straight_line_w_cos(t, y, p, theta):
    X1 = np.vstack((np.ones(len(t)), np.array(t), (t**p)*np.cos(t + theta))).T
    beta = np.linalg.lstsq(X1, y, rcond=None)[0]
    u = np.arange(0, 26.01, 0.01)
    v2 = np.vstack((np.ones(len(u)), np.array(u), (u**p)*np.cos(u + theta))).T␣
↪@ beta
    res = y - X1 @ beta
    resid = np.linalg.norm(res, 2)**2
    return resid, res, beta, v2

def fit_straight_line_w_cos_to_opt(variables, constants):
    X1 = np.vstack((np.ones(len(constants[0])), np.array(constants[0]),
                    (constants[0]**variables[0])*np.cos(constants[0] +␣
↪variables[1]))).T
    beta = np.linalg.lstsq(X1, constants[1], rcond=None)[0]
    u = np.arange(0, 26.01, 0.01)
    v2 = np.vstack((np.ones(len(u)), np.array(u), (u**variables[0])*np.cos(u +␣
↪variables[1]))).T @ beta
    res = constants[1] - X1 @ beta
    resid = np.linalg.norm(res, 2)**2
    return resid

variables = [1, 1] # initial guess for p and theta
constants = [t_dropped, y_dropped] # format for optimisation

p, theta = minimize(fit_straight_line_w_cos_to_opt, variables, args=constants).x

print("Optimum p: {}, and Optimum theta: {}".format(p, theta))

resid, res4, beta, v3 = fit_straight_line_w_cos(t_dropped, y_dropped, p, theta)

u = np.arange(0, 26.01, 0.01)

plt.figure()
plt.plot(t_dropped, y_dropped, 'ro')
plt.plot(u, v1, 'b-', u, v2, 'g-', u, v3, 'y-')
plt.title('Plot of Models against Real Data: original (blue), sqrt(t)*sin(t)␣
 ↪(green), (t**p)*cos(t + theta) (yellow)')
plt.show()
```

```
Optimum p: 2.98124476102927, and Optimum theta: 0.29086734595916О3
```



```
[ ]:
```

```
[ ]:
```