- 1.

  (a.)

$$\frac{\partial E2}{\partial b} = \sum_{i=1}^{n} 2(y_i - (m_i + b))(-1) = 0$$

$$\frac{\sum_{i=1}^{n} y_i - m\sum_{i=1}^{n} x_i}{n} = b$$

$$\frac{\partial E2}{\partial m} = \sum_{i=1}^{n} 2(y_i - (m_i + b))(-x_i) = 0$$

$$\frac{\sum_{i=1}^{n} x_i y_i - b\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i^2} = m$$

  Given the datasets, we could synthesize this system of equations and get the values for m and b.

  - (b)The same equations with the coefficient terms shifted to one side give the following matrix solution:

$$\begin{bmatrix} n & -\sum_{i=1}^{n} x_i \\ -\sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$

  - I would expect that if I knew how, these systems would simplify into each other. This is not surprising as both projections and critical points serve to optimize the concept of error from different angles.

- 4. This systems yields the following augmented matrix which we row reduce:

$$\begin{bmatrix} 1 & 3 & 2 & 6 \\ 2 & 5 & -1 & 6 \\ -4 & 0 & 7 & 3 \\ 9 & 8 & -2 & 15 \\ 3 & 3 & 8 & 14 \\ -1 & 2 & -4 & -3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

  Here we see that the system is consistent as our $A[1:3]$ has pivot columns. We get

$$x_1 = x_2 = x_3 = 1$$

  Our Normal equations provides

$$\begin{bmatrix} 112 & 92 & -18 \\ 92 & 111 & 1 \\ -18 & 1 & 138 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 186 \\ 204 \\ 121 \end{bmatrix}$$

  which resolves to the same solution.

2DeLay_comp3

April 26, 2021

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
x = np.array([2,5,8,10,13,19,30])
y = np.array([1,7,0,10,-13,-21,-32])
y = np.transpose(y)
x.size
```

[1]: 7

Create an array of terms excluding the parameters we're estimating.

[2]:
```python
A = np.array([[1, x[0]],
              [1, x[1]],
              [1, x[2]],
              [1, x[3]],
              [1, x[4]],
              [1, x[5]],
              [1, x[6]]
             ])
```

Build $A^T A$ and $A^T y$

[3]:
```python
AtA = np.matmul(np.transpose(A),A)
Aty = np.matmul(np.transpose(A),y)
```

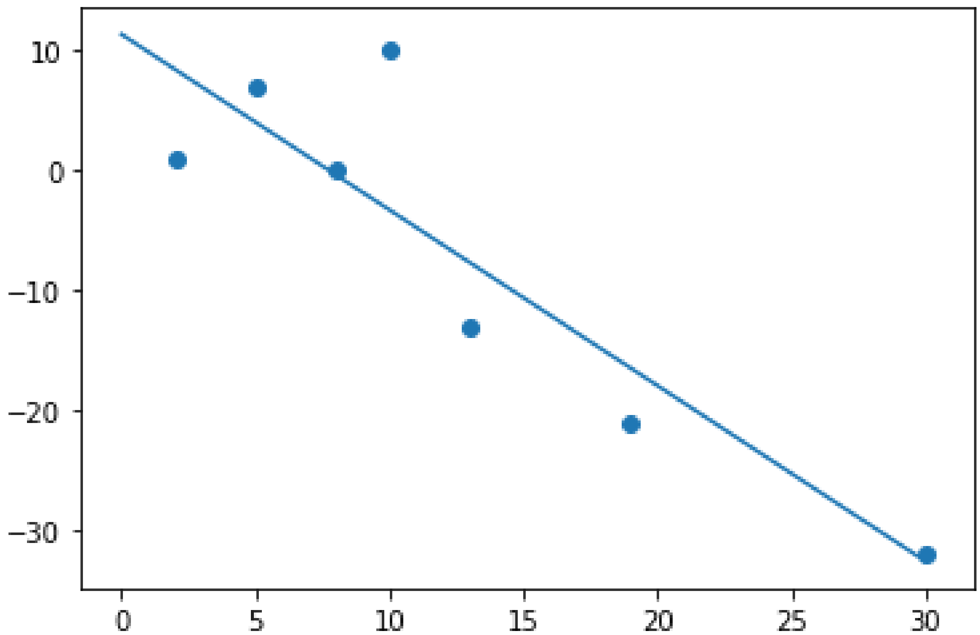
Estimate c_i.

[4]:
```python
coeffs = np.linalg.solve(AtA, Aty)
print(coeffs)
```

```
[11.36946203 -1.46650844]
```

Here we have the the curve of best fit given $\{x_i\}$ and $f(x)$.

[5]:
```python
xTest = np.linspace(0,30,100)
func = lambda x: coeffs[0]+coeffs[1]*x
plt.plot(xTest,func(xTest))
plt.scatter(x,y)
plt.show()
```


10
0
−10
−20
−30
0
5
10
15
20
25
30

[ ]:

DeLay_comp3

April 26, 2021

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi, 3*np.pi/2, 2*np.pi])
y = np.array([0,7,-5,6,1,-3,-1])
x.size
```

[8]:
```
7
```

Create an array of terms excluding the parameters we're estimating.

[9]:
```
A = np.array([[1, np.sin(x[0]), np.sin(2*x[0]), np.sin(3*x[0])],
         [1, np.sin(x[1]), np.sin(2*x[1]), np.sin(3*x[1])],
         [1, np.sin(x[2]), np.sin(2*x[2]), np.sin(3*x[2])],
         [1, np.sin(x[3]), np.sin(2*x[3]), np.sin(3*x[3])],
         [1, np.sin(x[4]), np.sin(2*x[4]), np.sin(3*x[4])],
         [1, np.sin(x[5]), np.sin(2*x[5]), np.sin(3*x[5])],
         [1, np.sin(x[6]), np.sin(2*x[6]), np.sin(3*x[6])]])
```

Build A^TA and A^Ty

[10]:
```
AtA = np.matmul(np.transpose(A),A)
y = np.array([
    [0],
    [7],
    [-5],
    [6],
    [1],
    [-3],
    [-1]]
)
Aty = np.matmul(np.transpose(A),y)
```

Estimate c_i.

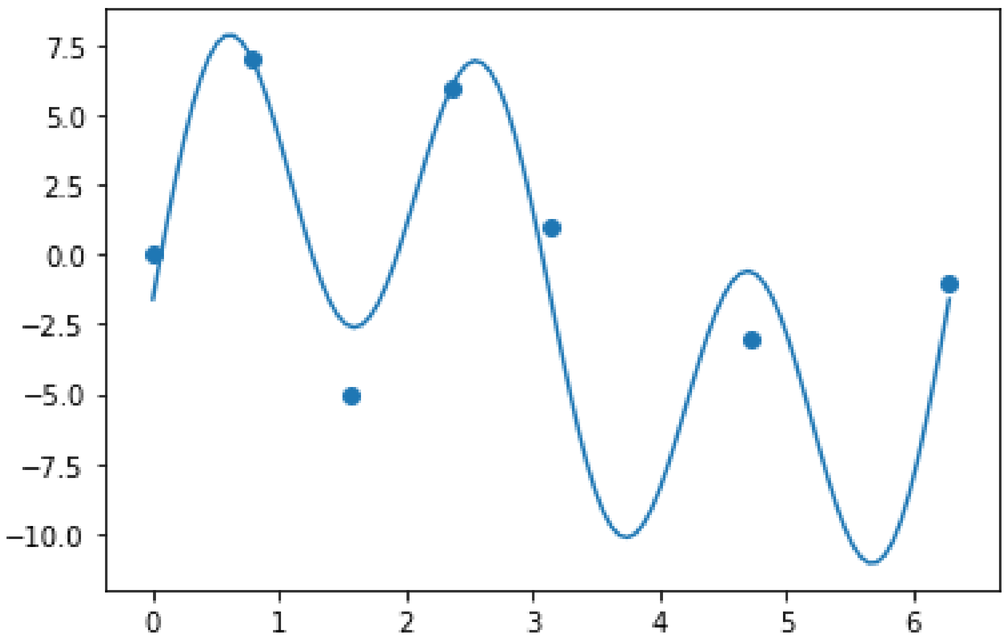
[11]:
```
coeffs = np.linalg.solve(AtA, Aty)
print(coeffs)
```

```
[[-1.6       ]
 [ 5.22756493]
```

```
[ 0.5        ]
[ 6.22756493]]
```

Here we have the the curve of best fit given $\{x_i\}$ and $f(x)$.

[15]:
```
xTest = np.linspace(0,2*np.pi,100)
func = lambda x: coeffs[0]+coeffs[1]*np.sin(x)+coeffs[2]*np.
↪sin(2*x)+coeffs[3]*np.sin(3*x)
plt.plot(xTest,func(xTest))
plt.scatter(x,y)
plt.show()
```



[ ]:
```
```

5DeLay_comp3

April 26, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_excel('popdata.xlsx')

x = np.reshape(np.array(df.decade),(df.shape[0],1))
rawy = np.reshape(np.array(df.raw),(df.shape[0],1))
y = np.log(rawy)
```

Create an array of terms excluding the parameters we're estimating.

[2]:
```
ones = np.ones((df.shape[0],1))
A = np.hstack((ones,x))
```

Build $A^T A$ and $A^T y$

[3]:
```
AtA = np.matmul(np.transpose(A),A)

Aty = np.matmul(np.transpose(A),y)
```

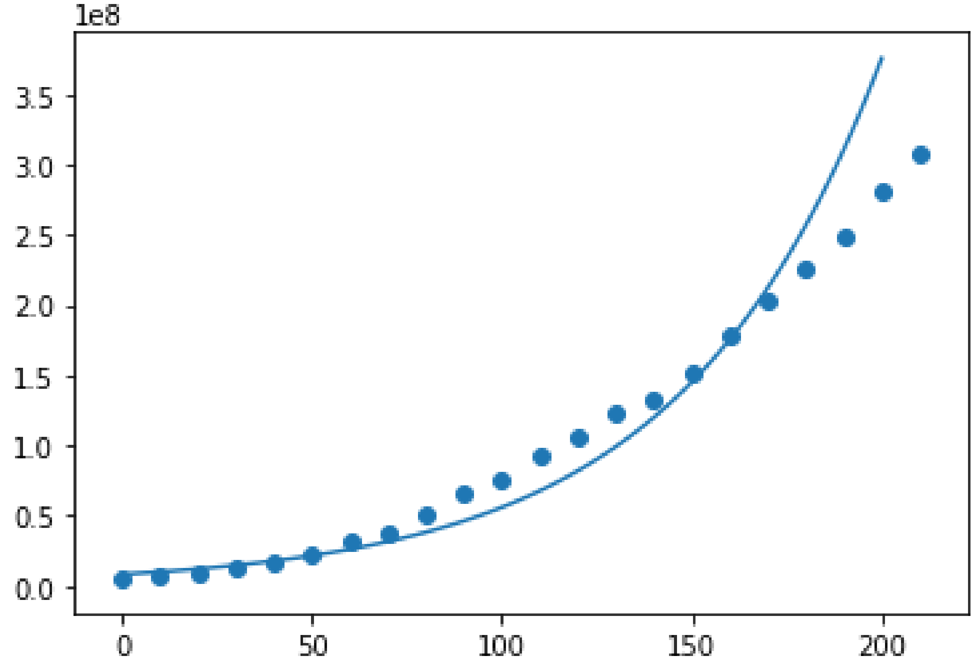
Estimate c_i.

[4]:
```
coeffs = np.linalg.solve(AtA, Aty)
print(coeffs)
```

```
[[15.94253018]
 [ 0.0190097 ]]
```

Here we have the the curve of best fit given $\{x_i\}$ and $f(x)$.

[5]:
```
xTest = np.linspace(0,200,100)
func = lambda x: np.exp(coeffs[0])*np.exp(coeffs[1]*x)
plt.plot(xTest,func(xTest))
plt.scatter(x,rawy)
plt.show()
```


1e8
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200

[ ]: