Applied Mathematics Assignment 02
TW244 Applied Differential Equations

https://github.com/BhekimpiloNdhlela/TW244AppliedDifferentialEquations

Bhekimpilo Ndhlela (18998712)

22 August 2018

NOTE: PLEASE REFER TO THE LAST PAGE FOR UTILITY FUNCTIONS.

Problem 1

Question 1a.)

Python Source Code For Question 1a.)

```
Y = np.arange(1790, 1960, 10)
P = np.array([3.929,  5.308,   7.240,   9.368,   12.866, 17.069,\
              23.192, 31.433,  38.558,  50.156,  62.948, 75.996,\
              91.972, 105.711, 122.775, 131.669, 150.697])
year_pop = {year: population for year, population in zip(Y, P)}
Q = lambda t: 1.0/10.0 * (year_pop[t+10]/year_pop[t]   1)
Qt = np.array([Q(Y[i]) for i in xrange(len(Y)   1)])
plot4q1a(Y, P, Qt)
```

t	$P(t)$	$Q(t)$
0	3.92900	0.03510
10	5.30800	0.03640
20	7.24000	0.02939
30	9.36800	0.03734
40	12.86600	0.03267
50	17.06900	0.03587
60	23.19200	0.03553
70	31.43300	0.02267
80	38.55800	0.03008
90	50.15600	0.02550
100	62.94800	0.02073
110	75.99600	0.02102
120	91.97200	0.01494
130	105.71100	0.01614
140	122.77500	0.00724
150	131.66900	0.01445

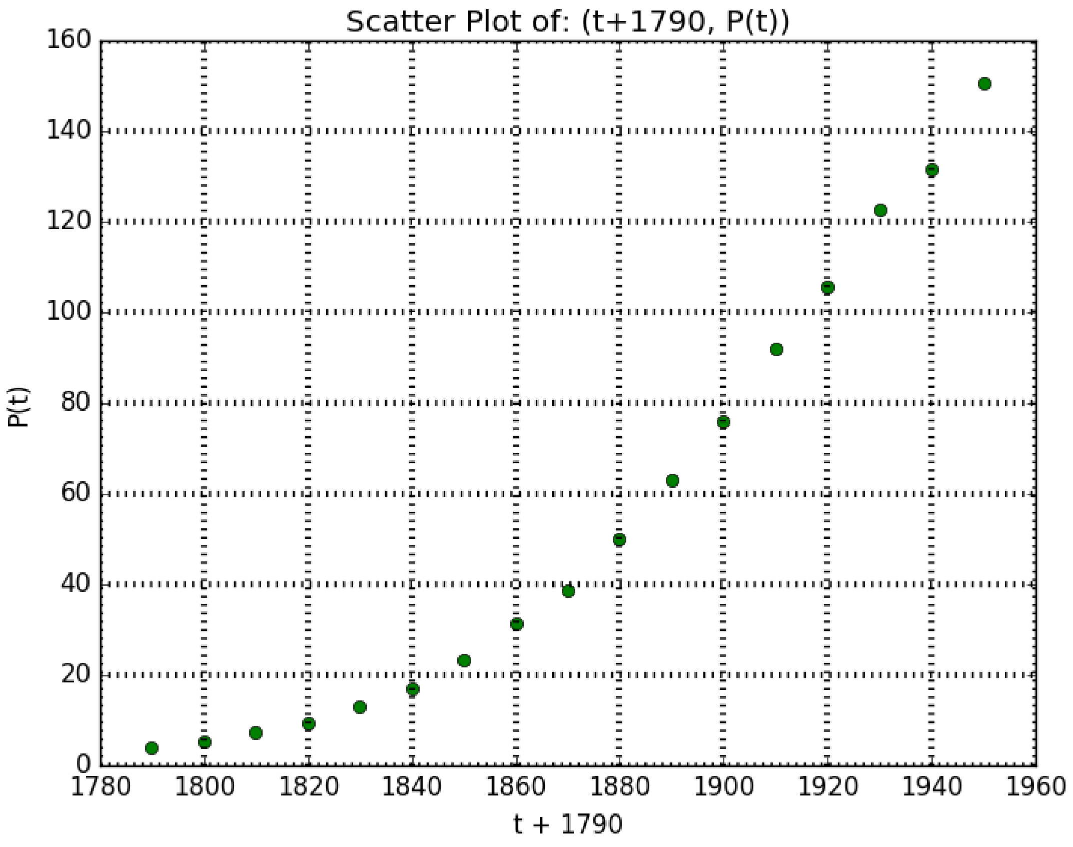

Figure 1: Scatter Plot of: $(t + 1790,\ P(t)))$

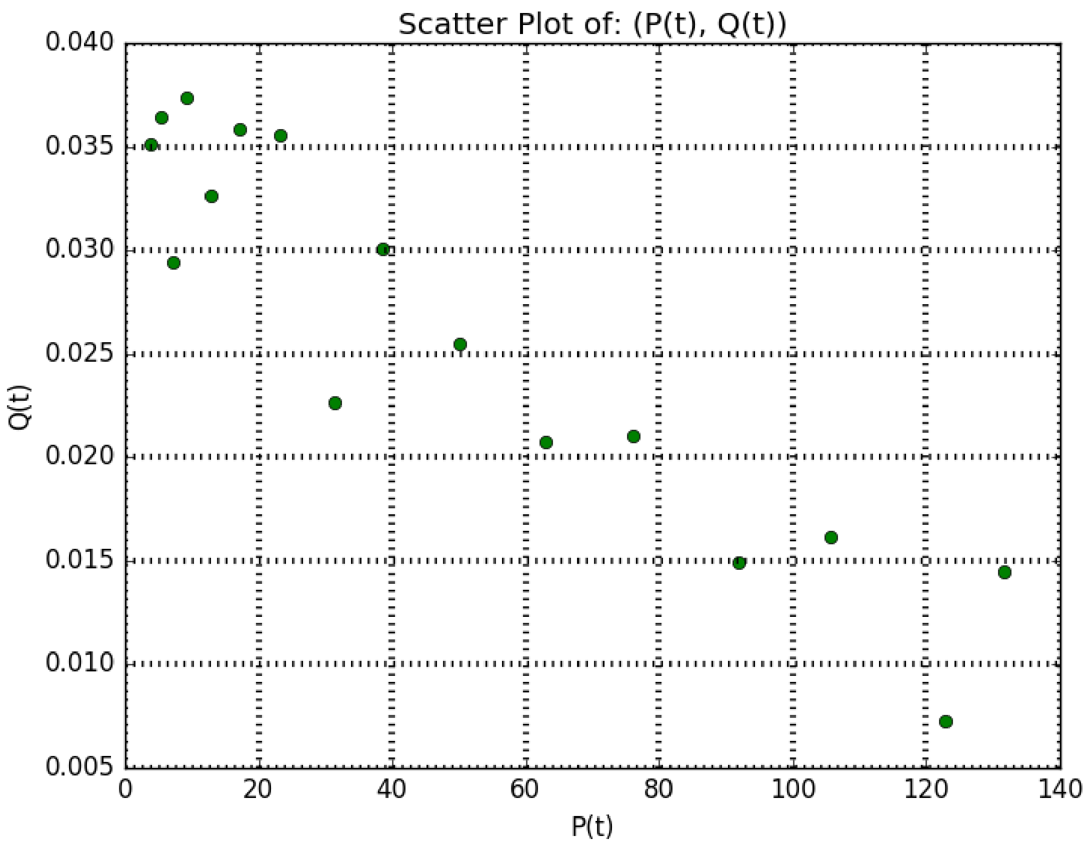

Figure 2: Scatter Plot of: $(P(t), Q(t))$

Question 1b.)

Python Source Code For Question 1b.)

```
Y = np.arange(1790, 1960, 10)
P = np.array([3.929,  5.308,   7.240,   9.368,   12.866, 17.069,\
              23.192, 31.433,  38.558,  50.156,  62.948, 75.996,\
              91.972, 105.711, 122.775, 131.669, 150.697])
year_pop = {year: population for year, population in zip(Y, P)}
Q = lambda t: 1.0/10.0 * (year_pop[t+10]/year_pop[t]   1)
Qt = np.array([Q(Y[i]) for i in xrange(len(Y)   1)])
a, b = get_coef(P[:len(Qt)], Qt, 1)
plot4q1b(P[:len(Qt)], Qt, a, b)
```

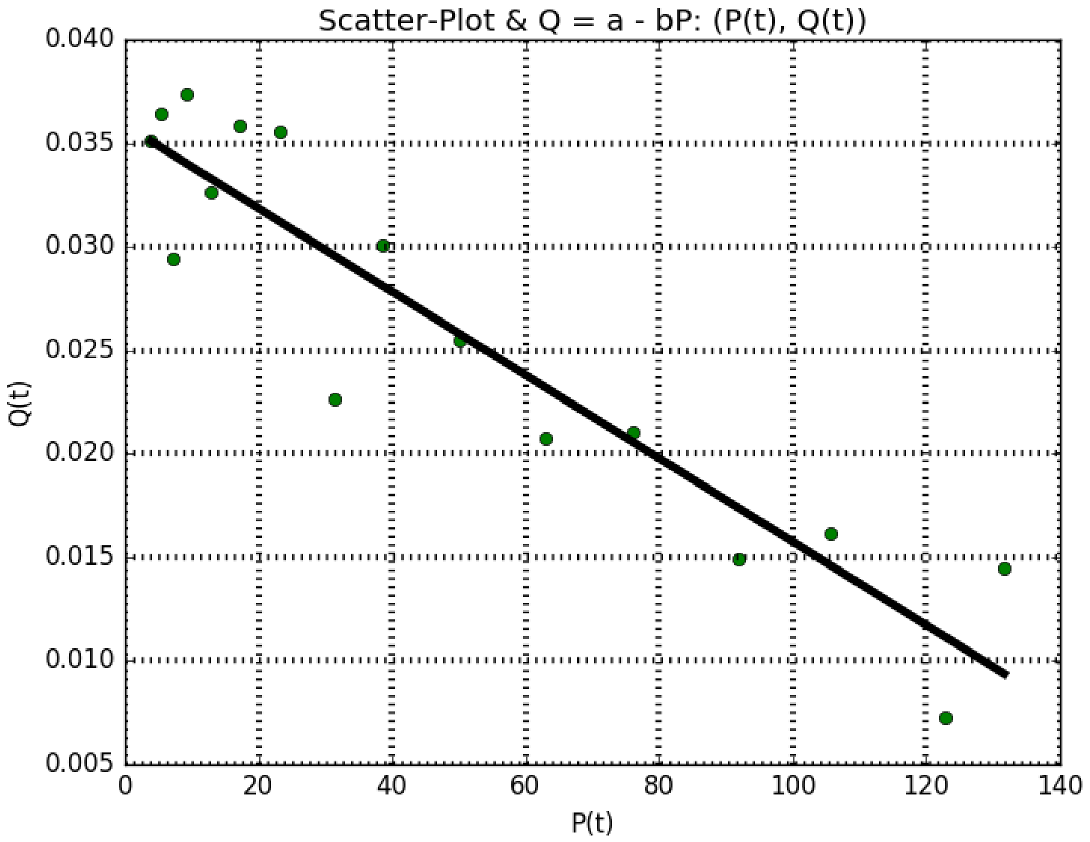

Figure 3: Scatter-Plot and $Q = a - bP$: $(P(t), Q(t))$

Question 1c.)

Python Source Code For Question 1c.)

```
Y = np.arange(1790, 1960, 10)
P = np.array([3.929,  5.308,   7.240,   9.368,   12.866, 17.069,\
              23.192, 31.433,  38.558,  50.156,  62.948, 75.996,\
              91.972, 105.711, 122.775, 131.669, 150.697])
year_pop = {year: population for year, population in zip(Y, P)}
Q = lambda t: 1.0/10.0 * (year_pop[t+10]/year_pop[t]   1)
Qt = np.array([Q(Y[i]) for i in xrange(len(Y)   1)])
a, b = get_coef(P[:len(Qt)], Qt, 1)
T = np.arange(0, 240, 10)
Pt = lambda t: (a*P[0])/(b*P[0]+(a b*P[0])*np.exp( a*t))
p1 = np.array([Pt(t) for t in T])
plot4q1c(Y, P, T + 1790, p1)
```

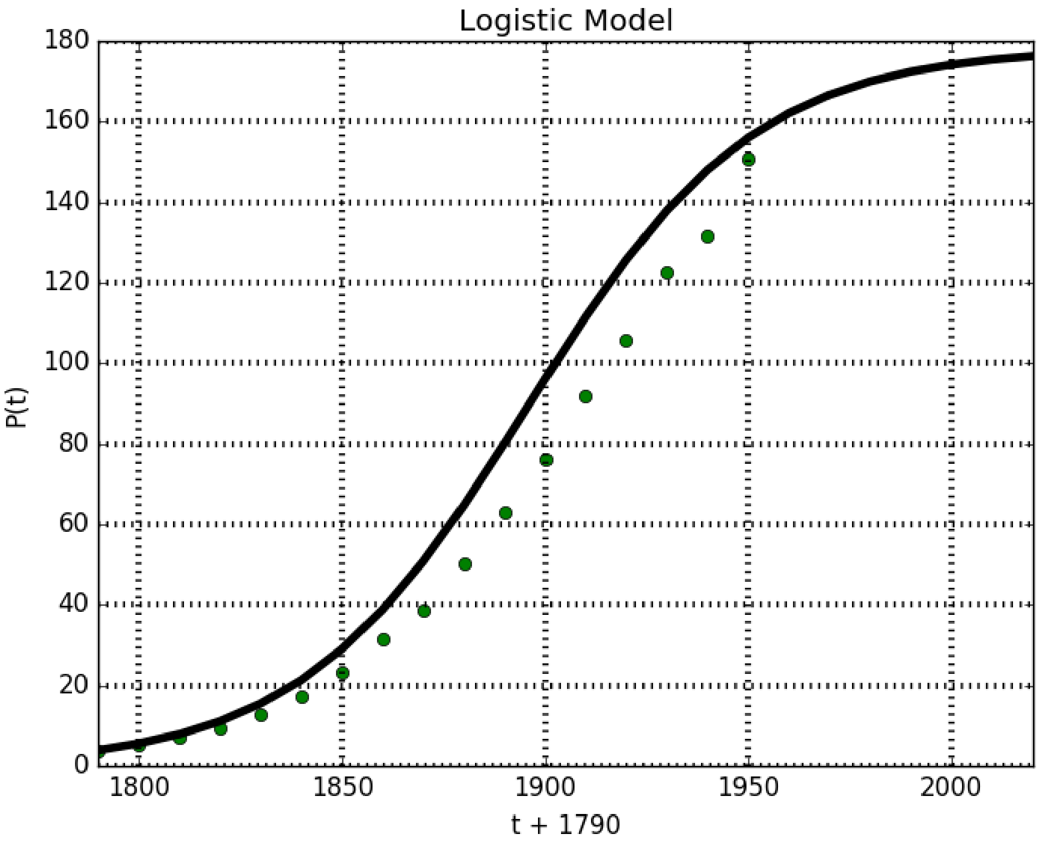

Figure 4: Logistic Model

Question 1d.)

The Logistic Model(My Model) predicts that the US population in 2018 is:

$P(2018) \approx 175.861$ **million (According to the Logistic Growth Model)**

However, according to Google the actual population of the US is:

$P(2018) \approx 326,766,748$ **million**

According to the Logistic Model the carrying capacity of the US is given by :

$$\lim_{t\to\infty} P(t) = \frac{a}{b}$$

But:

$$a = 0.0358837726755 \textbf{ and } b = 0.000201295462009$$

$$\Longrightarrow \lim_{t\to\infty} P(t) = \frac{a}{b} = 178.264190943 \textbf{ Million}$$

Remarks: I do not think that the **Logistic model** is a good model for this data because:

- By comparing the US population for 2018 computed by the logistic model and the Actual Population of the US. The population computed by the Logistic Model is almost two times smaller than the actual population.
- Also The Carrying capacity is supposed to be the Peak Population of US but according to Google, the carrying capacity is almost two(2) time smaller than the actual US population.

Problem 2

Question 2a.)

Required to make use of separation of variables to solve:

$$\frac{dP}{dt} = P(a - b\ln P)$$

and to show that:

$$P(t) = e^{\frac{a}{b} + Ce^{-bt}}$$

$$\frac{dP}{P(a - b\ln P)} = dt$$

$$\int \frac{dP}{P(a - b\ln P)} = \int dt$$

$$\implies \int \frac{dP}{P(a - b\ln P)} = t + C \textbf{ Where: } C \in \mathbb{R}$$

$$\textbf{Let: } P = e^u \implies dP = e^u du \implies dP = P du$$

$$\int \frac{du}{a - bu} = t + C$$

$$-\frac{1}{b}\ln(a - bu) = t + C$$

$$\ln(a - bu) = -bt + C$$

$$a - bu = Ce^{-bt}$$

$$u = Ce^{-bt} + \frac{a}{b}$$

$$\textbf{But: } P = e^u \implies P = e^{\frac{a}{b} + Ce^{-bt}}$$

$$\textbf{Hence: } P(t) = e^{\frac{a}{b} + Ce^{-bt}}$$

By the use of the fact that: $P(0) = P_0 = 3.929$ The following **holds true,** $wheneliminatingC$

$$\therefore P(0) = P_0 = e^{\frac{a}{b} + Ce^{-b(0)}} = e^{\frac{a}{b} + C}$$

$$\ln P_0 = \frac{a}{b} + C \implies C = \ln P_0 - \frac{a}{b}$$

$$\textbf{But: } P_0 = 3.929 \therefore C = \ln 3.929 - \frac{a}{b}$$

$$\therefore P(t) = e^{\frac{a}{b} + Ce^{-bt}} = e^{\frac{a}{b} + (\ln 3.929 - \frac{a}{b})e^{-bt}}$$

Question 2b.)

Python Source Code For Question 2b.)

```
Y = np.arange(1790, 1960, 10)
P = np.array([3.929,  5.308,   7.240,   9.368,   12.866, 17.069,\
              23.192, 31.433,  38.558,  50.156,  62.948, 75.996,\
              91.972, 105.711, 122.775, 131.669, 150.697])
              year_pop = {year: population for year, population in zip(Y, P)}
Q = lambda t: 1.0/10.0 * (year_pop[t+10]/year_pop[t]   1)
Qt = np.array([Q(Y[i]) for i in xrange(len(Y)   1)])
a, b = get_coef(P[:len(Qt)], Qt, 1)
P2 = np.log(P)
a2, b2 = get_coef(P2[:len(Qt)], Qt, 1)
plot4q2b(P[:len(Qt)], Qt, P2[:len(Qt)], a, b, a2, b2)
```

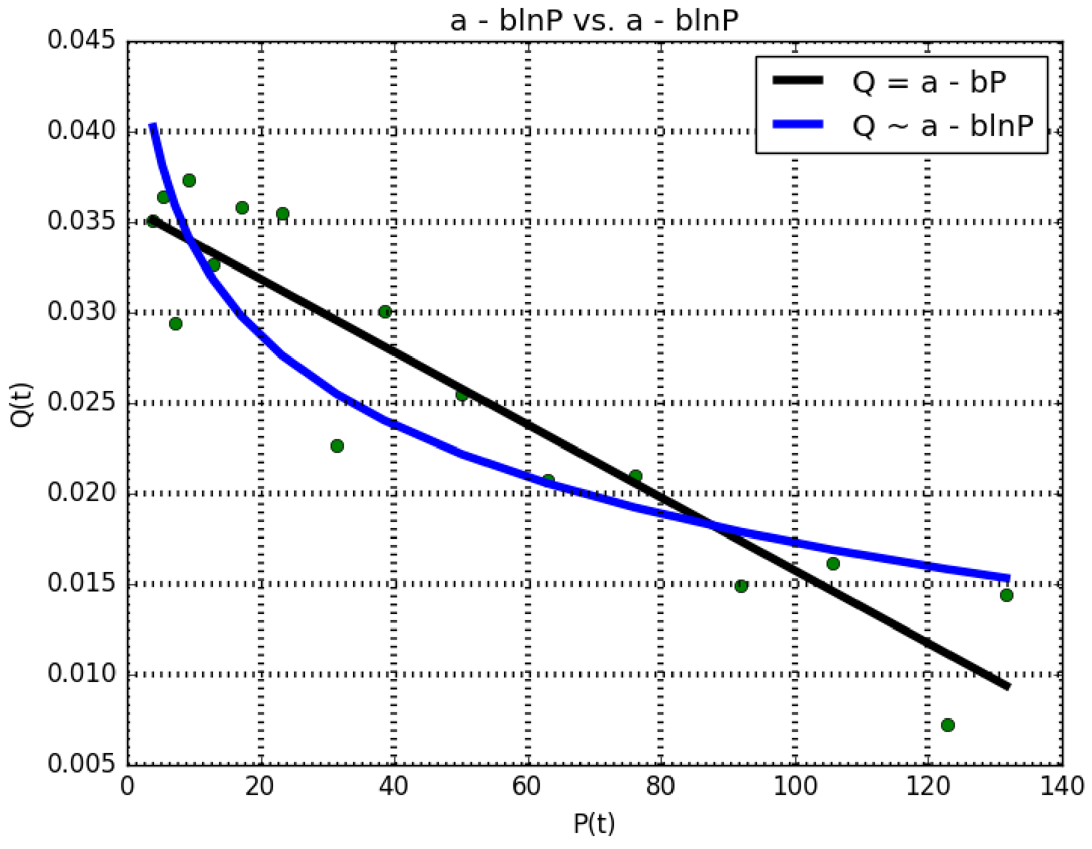

Figure 5: $Q \approx a - blnP$ vs.$Q = a - bP$

Question 2c.)

subsection*Python Source Code For Question 2c.)

```
Y = np.arange(1790, 1960, 10)
P = np.array([3.929,  5.308,   7.240,   9.368,   12.866, 17.069,\
              23.192, 31.433,  38.558,  50.156,  62.948, 75.996,\
              91.972, 105.711, 122.775, 131.669, 150.697])
              year_pop = {year: population for year, population in zip(Y, P)}
Q = lambda t: 1.0/10.0 * (year_pop[t+10]/year_pop[t]   1)
Qt = np.array([Q(Y[i]) for i in xrange(len(Y)   1)])
a, b = get_coef(P[:len(Qt)], Qt, 1)
P2 = np.log(P)
a2, b2 = get_coef(P2[:len(Qt)], Qt, 1)
c = np.log(3.929) (a2/b2)
Pt = lambda t : np.exp((a2/b2)+c*np.exp( b2*t))
p2 = np.array([Pt(t) for t in T])
plot4q2c(Y, P, T + 1790, p1, p2)
```

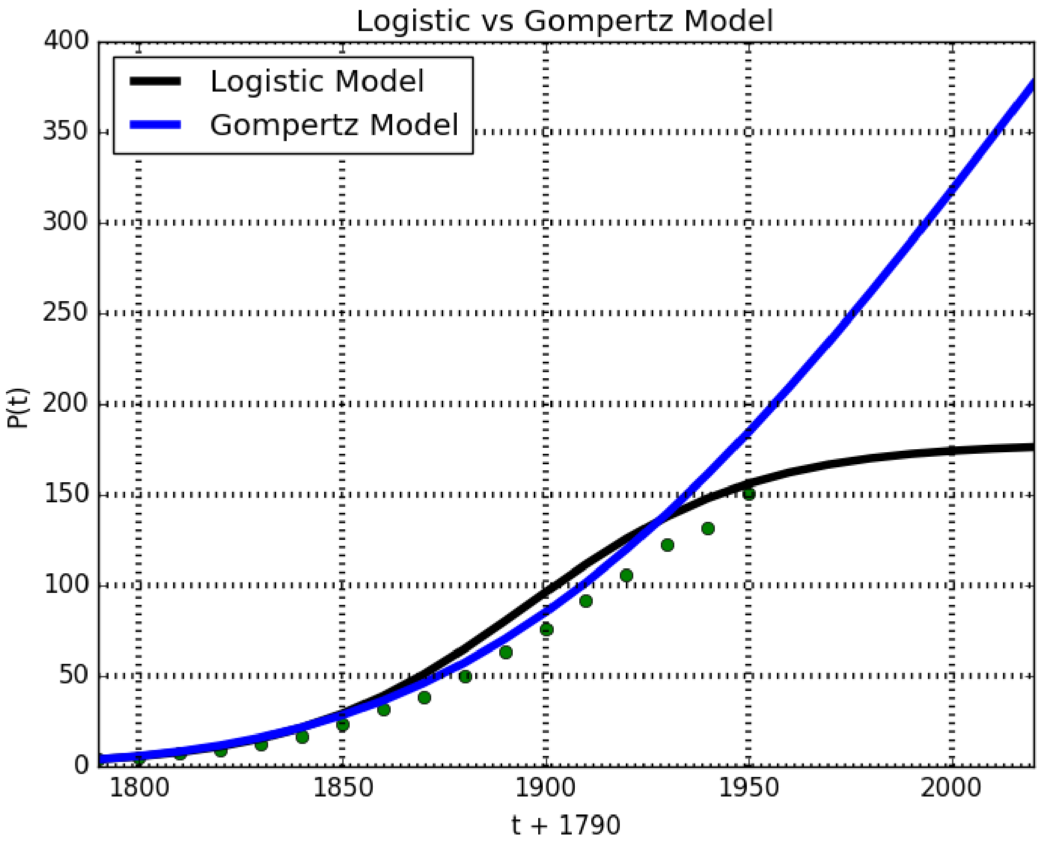

Figure 6: Logistic Model Vs. Gompertz Model

Question 2d.)

The limiting capacity (i.e., as $t \to \infty$) is:

$$\lim_{t\to\infty} P(t) = \lim_{t\to\infty} e^{\frac{a}{b}+Ce^{-bt}}$$

$$\implies e^{\frac{a}{b}+C}$$

$$\textbf{But: } C = \ln P_0 - \frac{a}{c}$$

$$\therefore \textbf{this} \implies \lim_{t\to\infty} P(t) = e^{\frac{a}{b}+\ln P_0 - \frac{a}{b}}$$

$$\implies \lim_{t\to\infty} P(t) = e^{\frac{a}{b}} e^{\ln P_0} e^{-\frac{a}{b}} = e^{\frac{a}{b}-\frac{a}{b}} P_0$$

$$\therefore \textbf{this} \implies \lim_{t\to\infty} P(t) = P_0 = 3.929$$

The limiting capacity/ carrying capacity of the:

Gompertz Model is, $\lim_{t\to\infty} P(t) = P_0 = 3.929$ **Million**

Logistic Model is, $\lim_{t\to\infty} P(t) = \frac{a}{b} \approx 178.264190943$ **Million**

Actual Population of USA at 2018**:** $P(2018) \approx 326,766,748$
Population of USA according to the Logistic Model at 2018**:** $P(2018) \approx 177$ **Million**
Population of USA according to the Gompertz Model at 2018**:** $P(2018) \approx 370$ **Million**

Remarks: The **Gompertz Model** is **more accurate** compared to the **Logistic Model** for modeling the US population because when one compares the currant Population (2018) of the US and the one that was modeled by both the Logistic Model and the Gompertz Model it is evident that the Gompertz model is more closer to the actual data or population than the Logistic Model.

However, when we compare the **limiting capacity/ Carrying Capacity (i.e., as** $t \to \infty$ **)** of both the Logistic and the Gompertz Model.
The Logistic Model makes more sense than the carrying capacity of the Gompertz which states that the Limiting Capacity is $P_0 \approx 3.929$ Million which is not realistic according to Population Biologists.

Question 2e.)

Given:

$$P(t) = e^{\frac{a}{b} + ce^{-bt}}$$

Required To Find:

$$t \textbf{ st. } P(t) = 600$$

$$\therefore 600 = e^{\frac{a}{b}} e^{ce^{-bt}}$$

$$\frac{600}{e^{\frac{a}{b}}} = e^{(\ln P_0 - \frac{a}{b})e^{-bt}}$$

$$\textbf{since: } C = \ln P_0 - \frac{a}{b}$$

$$\frac{600}{e^{\frac{a}{b}}} = e^{\ln P_0 e^{-bt} - \frac{a}{b} e^{-bt}}$$

$$\frac{600}{e^{\frac{a}{b}}} = \frac{e^{\ln P_0 e^{-bt}}}{e^{\frac{a}{b}} e^{-bt}} = \frac{{P_0}^{e^{-bt}}}{e^{\frac{a}{b}} e^{-bt}}$$

$$\frac{600}{e^{\frac{a}{b}}} = (\frac{P_0}{e^{\frac{a}{b}}})^{e^{-bt}}$$

$$\ln \frac{600}{e^{\frac{a}{b}}} = \ln (\frac{P_0}{e^{\frac{a}{b}}})^{e^{-bt}}$$

$$\ln \frac{600}{e^{\frac{a}{b}}} = e^{-bt} \ln \frac{P_0}{e^{\frac{a}{b}}}$$

$$\frac{\ln \frac{600}{e^{\frac{a}{b}}}}{\ln \frac{P_0}{e^{\frac{a}{b}}}} = e^{-bt}$$

$$\frac{1}{b} \ln [\frac{\ln \frac{P_0}{e^{\frac{a}{b}}}}{\ln \frac{600}{e^{\frac{a}{b}}}}] = t$$

But:

$$a = 0.049951900508, \; b = 0.00709406133538 \textbf{ and } P_0 = 3.929$$

$$\therefore t \approx 306.607221532 \approx 307 \textbf{ years}$$

Note Some of the code lines where continued on the previous line by a (;), so that I could save paper. However, the same code would still execute.

Utility Functions for Question 1 and 2

```
def plot4q1a(Y, P, Q):
    plt.subplot(121)
    plt.title('Scatter␣Plot␣of:␣(t+1790,␣P(t))')
    plt.xlabel('t␣+␣1790'); plt.ylabel('P(t)')
    plt.plot(Y, P, 'og', linewidth=4)
    plt.grid(True, linewidth=3); plt.show()
    plt.title('Scatter␣Plot␣of:␣(P(t),␣Q(t))')
    plt.xlabel('P(t)'); plt.ylabel('Q(t)')
    plt.plot(P[:len(Q)], Q, 'og', linewidth=4)
    plt.grid(True, linewidth=3); plt.show()

def plot4q1b(x, y, a, b):
    plt.title('Scatter  Plot␣&␣Q␣=␣a␣ ␣bP:␣(P(t),␣Q(t))')
    plt.xlabel('P(t)'); plt.ylabel('Q(t)')
    plt.plot(x, y, 'og', linewidth=4)
    best_fit = np.array([a   b*x[i] for i in xrange(len(y))])
    plt.plot(x, best_fit, ' k', linewidth=4)
    plt.grid(True, linewidth=3); plt.show()

def plot4q1c(Y, P, x, y):
    plt.title('Logistic␣Model')
    plt.xlabel('t␣+␣1790'); plt.ylabel('P(t)'); plt.plot(Y, P, 'og', linewidth=4)
    plt.plot(x, y, ' k', linewidth=4); plt.xlim([1790, 2020])
    plt.grid(True, linewidth=3); plt.show()

def plot4q2b(x1, y1, x2, a1, b1, a2, b2):
    plt.title('Q␣=␣a␣ ␣bP␣vs.␣Q␣~␣a␣ ␣blnP')
    plt.xlabel('P(t)'); plt.ylabel('Q(t)')
    plt.plot(x1, y1, 'og', linewidth=4)
    best_fit1 = np.array([a1   b1*x1[i] for i in xrange(len(y1))])
    best_fit2 = np.array([a2   b2*x2[i] for i in xrange(len(y1))])
    plt.plot(x1, best_fit1, ' k', linewidth=4, label='Q␣=␣a␣ ␣bP')
    plt.plot(x1, best_fit2, ' b', linewidth=4, label='Q␣~␣a␣ ␣blnP')
    plt.legend(loc=0); plt.grid(True, linewidth=3); plt.show()

def plot4q2c(Y, P, x, y1, y2):
    plt.title('Logistic␣vs␣Gompertz␣Model')
    plt.xlabel('t␣+␣1790'); plt.ylabel('P(t)')
    plt.plot(Y, P, 'og', linewidth=4)
    plt.plot(x, y1, ' k', linewidth=4, label='Logistic␣Model')
    plt.plot(x, y2, ' b', linewidth=4, label='Gompertz␣Model')
    plt.xlim([1790, 2020]); plt.legend(loc=0); plt.grid(True, linewidth=3)
    plt.show()

def get_coef(x, y, deg):
    c = np.polyfit(x, y, deg)
    return c[1],  1*c[0] if (c[0] < 0) else c[0]
```