Population Growth 2

Dr. Murad A. Yaghi

- In the previous lesson, we simulated a model of the world population with constant growth.

- In this lesson, we'll see if we can make a better model with growth proportional to the population.

- The constant growth model is wrong for the following reasons:
  - It didn’t show great accuracy relative to the real data.
  - It is conceptually wrong: the growth resulting from 1 billion population in the next year shouldn’t be the same growth resulting from 2 billion.
- It is hard to imagine how people all over the world could conspire to keep population growth constant from year to year.

- If some fraction of the population dies each year, and some fraction gives birth, we can compute the net change in the population each year.

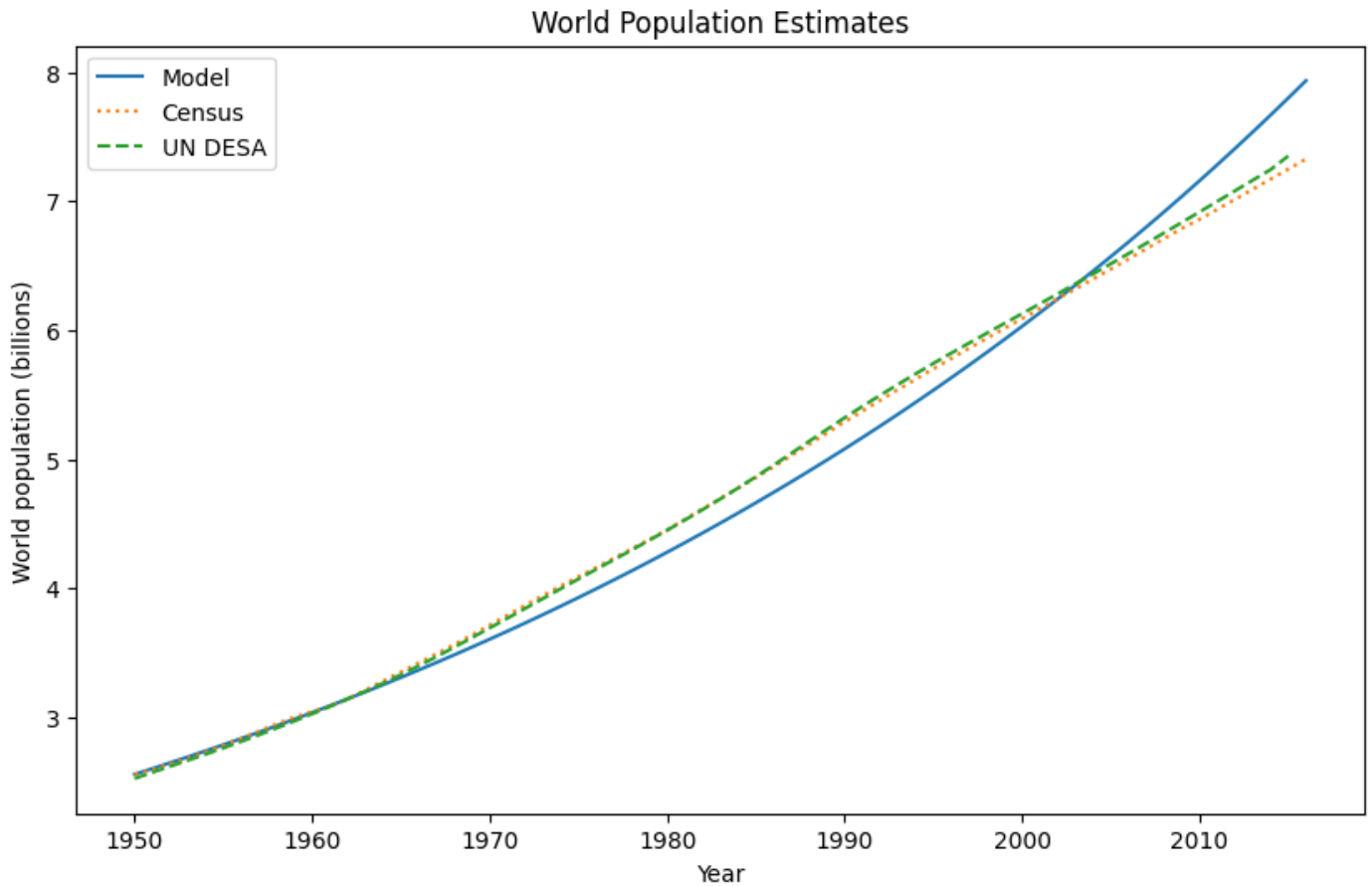
- We will define two parameters
  - birth_rate
  - death_rate

- Each time through the loop (update), we use the parameter birth_rate to compute the number of births, and death_rate to compute the number of deaths in the current population.

```python
class PopulationEstimateSystem:
    def __init__(self, p_0, birth_rate, death_rate):
        self.p_0 = p_0 # Initial population
        self.birth_rate = birth_rate # Birth rate per year as a proportion
        self.death_rate = death_rate # Death rate per year as a proportion
        self.population = p_0 # Current population, starts at initial value

    def update(self):
        growth = self.population * (self.birth_rate - self.death_rate)
        self.population = self.population + growth
        return self.population
```

```
t_0 = census.index[0]
t_end = census.index[-1]
p_0 = census[t_0]
birth_rate = 0.025
death_rate = 0.0077

# Create a PopulationEstimateSystem instance
population_system = PopulationEstimateSystem(p_0, birth_rate, death_rate)

population = pd.Series([p_0], index=[t_0]) # initialize the population Series
population[t_0] = population_system.population # Initialize with the initial population


for t in range(t_0, t_end):
    population[t+1] = population_system.update()
```

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(population.index, population, label='Model')
plt.plot(census.index, census, label='Census',
linestyle=':')
plt.plot(un.index, un, label='UN DESA', linestyle='--')
plt.xlabel('Year')
plt.ylabel('World population (billions)')
plt.title('World Population Estimates')
plt.legend()
plt.show()
```


World Population Estimates
Model
Census
UN DESA
World population (billions)
8
7
6
5
4
3
1950
1960
1970
1980
1990
2000
2010
Year

Task

- Compute mean absolute error
- Compute maximum absolute error
- Compute relative error

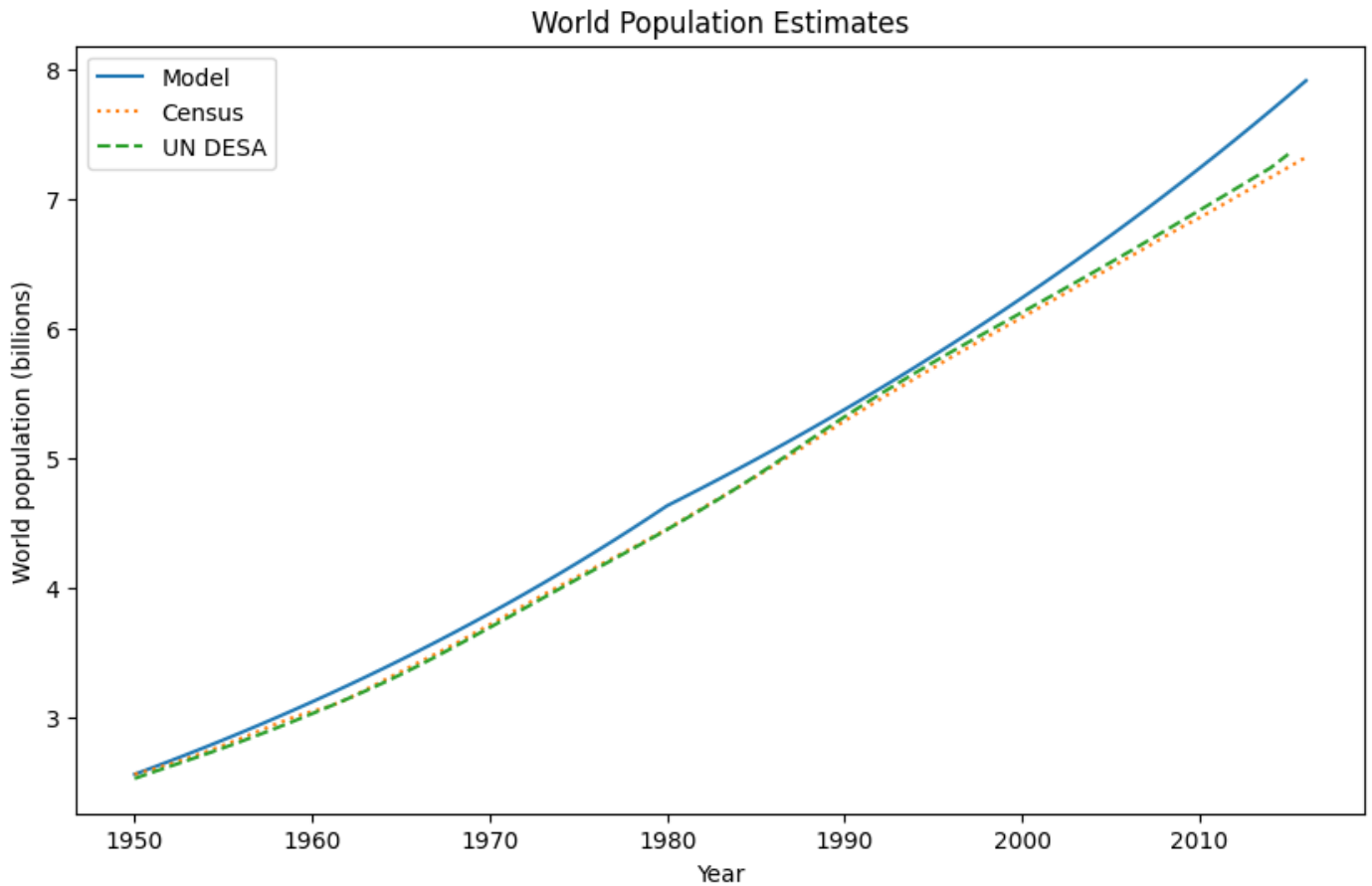
Exercise

- Maybe the reason for the previous model doesn't work very well is that the growth rate, is changing over time.
- So let's try a model with different growth rates before and after 1980 (as an arbitrary choice).
- The system update function should contain two parameters:
  - The growth rate before 1980, growth1.
  - The growth rate after 1980, growth2.
- It should use t to determine which growth rate to use.
- Test your function and adjust the parameters growth1 and growth2 to fit the data as well as you can.

```python
class PopulationEstimateSystem:
    def __init__(self, p_0, growth_rate1, growth_rate2):
        self.p_0 = p_0 # Initial population
        self.growth_rate1 = growth_rate1 # Growth rate before 1980
        self.growth_rate2 = growth_rate2 # Growth rate after 1980
        self.population = p_0 # Current population, starts at initial value

    def update(self, t):
        if t < 1980:
            growth_rate = self.growth_rate1
        else:
            growth_rate = self.growth_rate2

        growth = self.population * growth_rate
        self.population = self.population + growth

        return self.population
```

```
t_0 = census.index[0]
t_end = census.index[-1]
p_0 = census[t_0]
growth1 = 0.02 # Growth rate before 1980
growth2 = 0.015 # Growth rate after 1980

# Create a PopulationEstimateSystem instance
population_system = PopulationEstimateSystem(p_0, growth1, growth2)
population = pd.Series([p_0], index=[t_0]) # initialize the population Series
population[t_0] = population_system.population # Initialize with the initial population

for t in range(t_0, t_end):
        population[t+1] = population_system.update(t)
```

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(population.index, population, label='Model')
plt.plot(census.index, census, label='Census',
linestyle=':')
plt.plot(un.index, un, label='UN DESA', linestyle='--')
plt.xlabel('Year')
plt.ylabel('World population (billions)')
plt.title('World Population Estimates')
plt.legend()
plt.show()
```


World Population Estimates
Model
Census
UN DESA
World population (billions)
8
7
6
5
4
3
1950
1960
1970
1980
1990
2000
2010
Year

Task

- Compute mean absolute error
- Compute maximum absolute error
- Compute relative error

- We developed a population model where net growth during each time step is proportional to the current population.

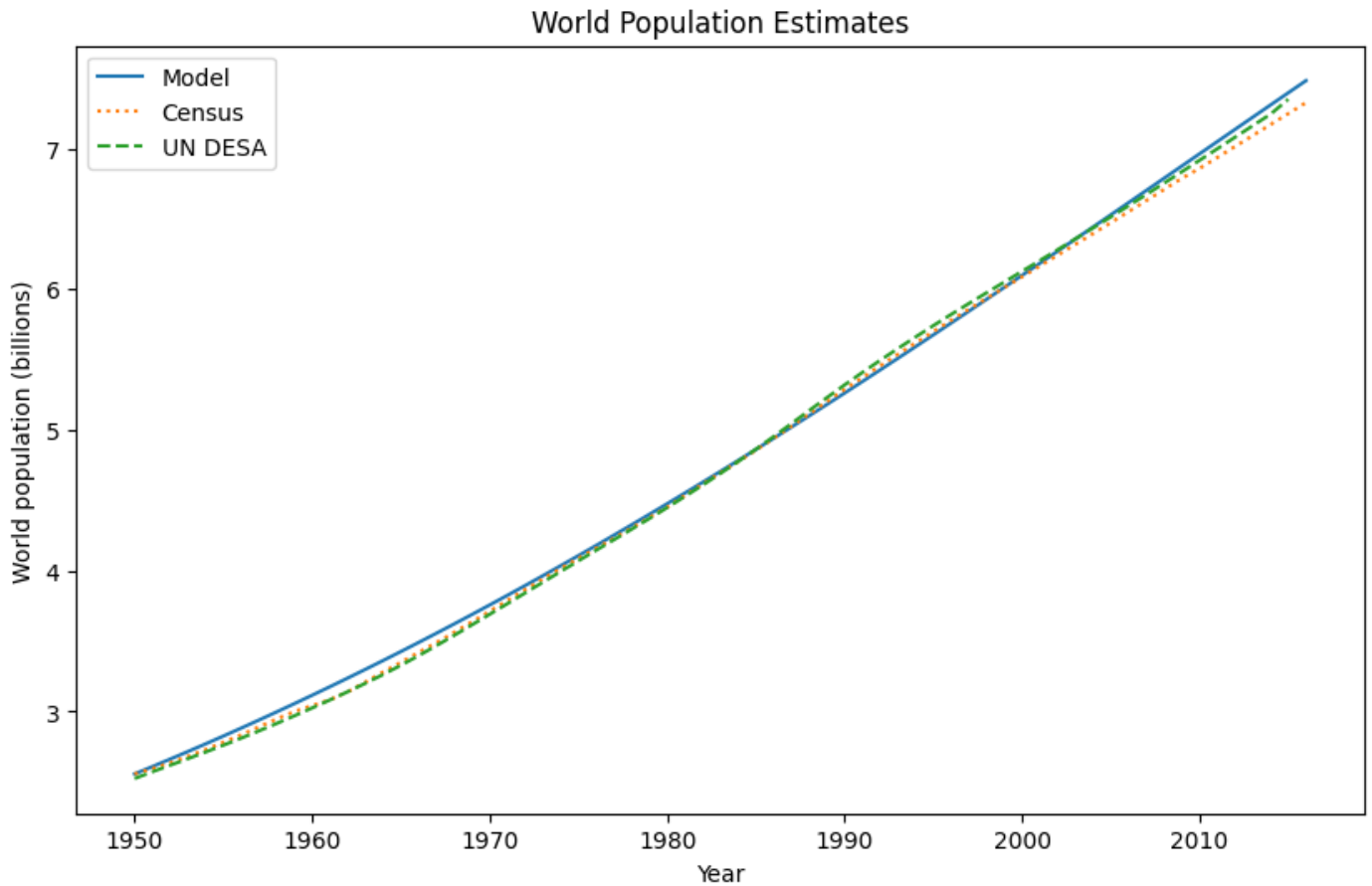
- This model seems more realistic than the constant growth model, but it does not fit the data as well.

- There are a few things we could try to improve the model:
  - Maybe net growth depends on the current population, but the relationship is quadratic, not linear.

```python
class PopulationEstimateSystem:
    def __init__(self, p_0, alpha, beta):
        self.p_0 = p_0 # Initial population
        self.alpha = alpha # Linear growth rate coefficient
        self.beta = beta # Quadratic growth rate coefficient
        self.population = p_0 # Current population, starts at the initial value

    def update(self):
        growth = self.alpha * self.population + self.beta * self.population**2
        self.population = self.population + growth
        return self.population
```

```
t_0 = census.index[0]
t_end = census.index[-1]
#t_end = 2100
p_0 = census[t_0]
alpha = 0.025
beta = -0.0018

# Create a PopulationEstimateSystem instance
population_system = PopulationEstimateSystem(p_0, alpha, beta)

population = pd.Series([p_0], index=[t_0]) # initialize the population Series
population[t_0] = population_system.population # Initialize with the initial population


for t in range(t_0, t_end):
        population[t+1] = population_system.update()
```

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(population.index, population, label='Model')
plt.plot(census.index, census, label='Census',
linestyle=':')
plt.plot(un.index, un, label='UN DESA', linestyle='--')
plt.xlabel('Year')
plt.ylabel('World population (billions)')
plt.title('World Population Estimates')
plt.legend()
plt.show()
```


World Population Estimates
Model
Census
UN DESA
World population (billions)
3
4
5
6
7
1950
1960
1970
1980
1990
2000
2010
Year

Task

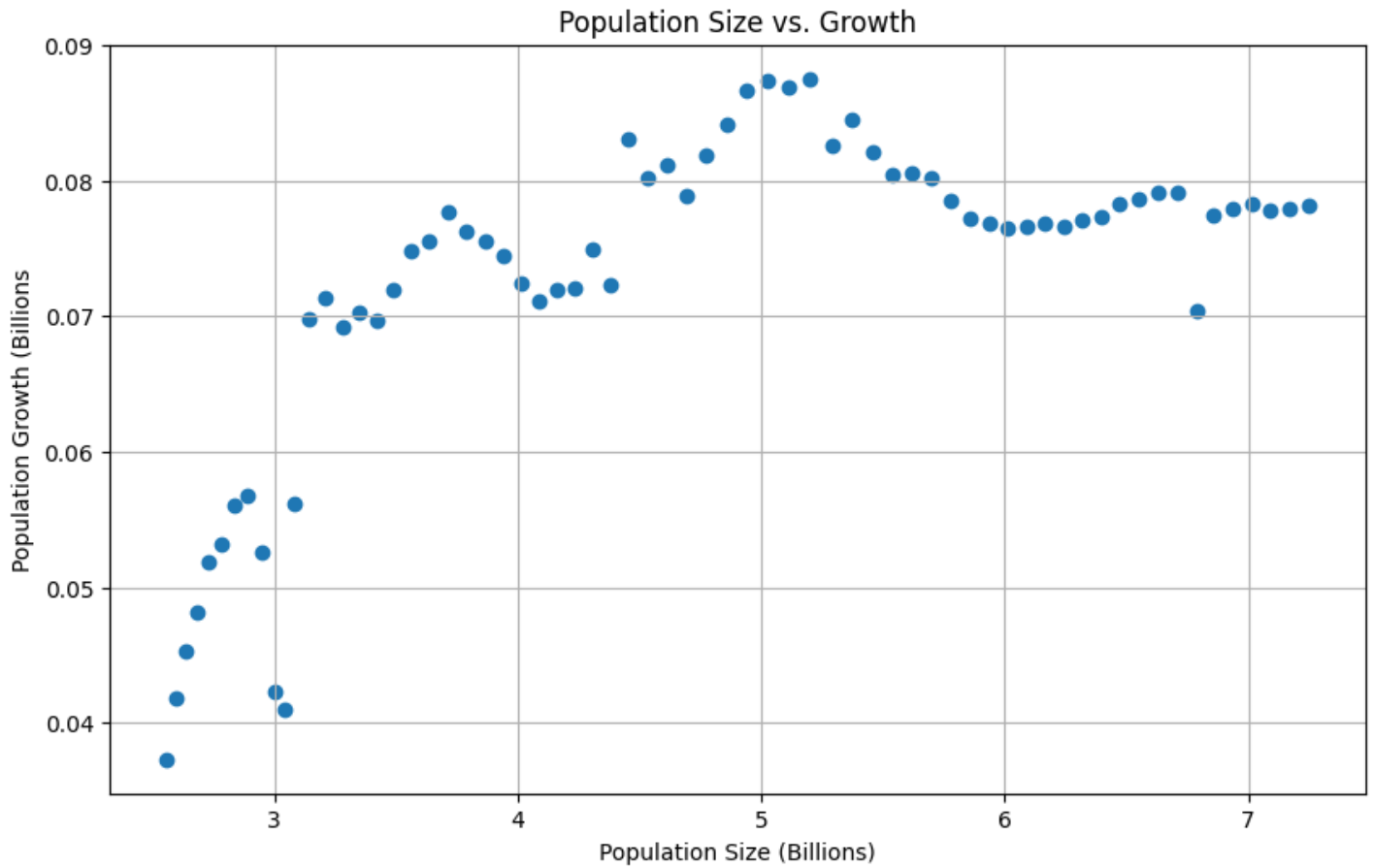
- Compute mean absolute error
- Compute maximum absolute error
- Compute relative error

- We have tried many functions for growth, are we supposed to try all functions that we know?
- Isn’t there any better way to know exactly how the growth function looks like?
- Can’t we just plot the growth from the empirical data we have?

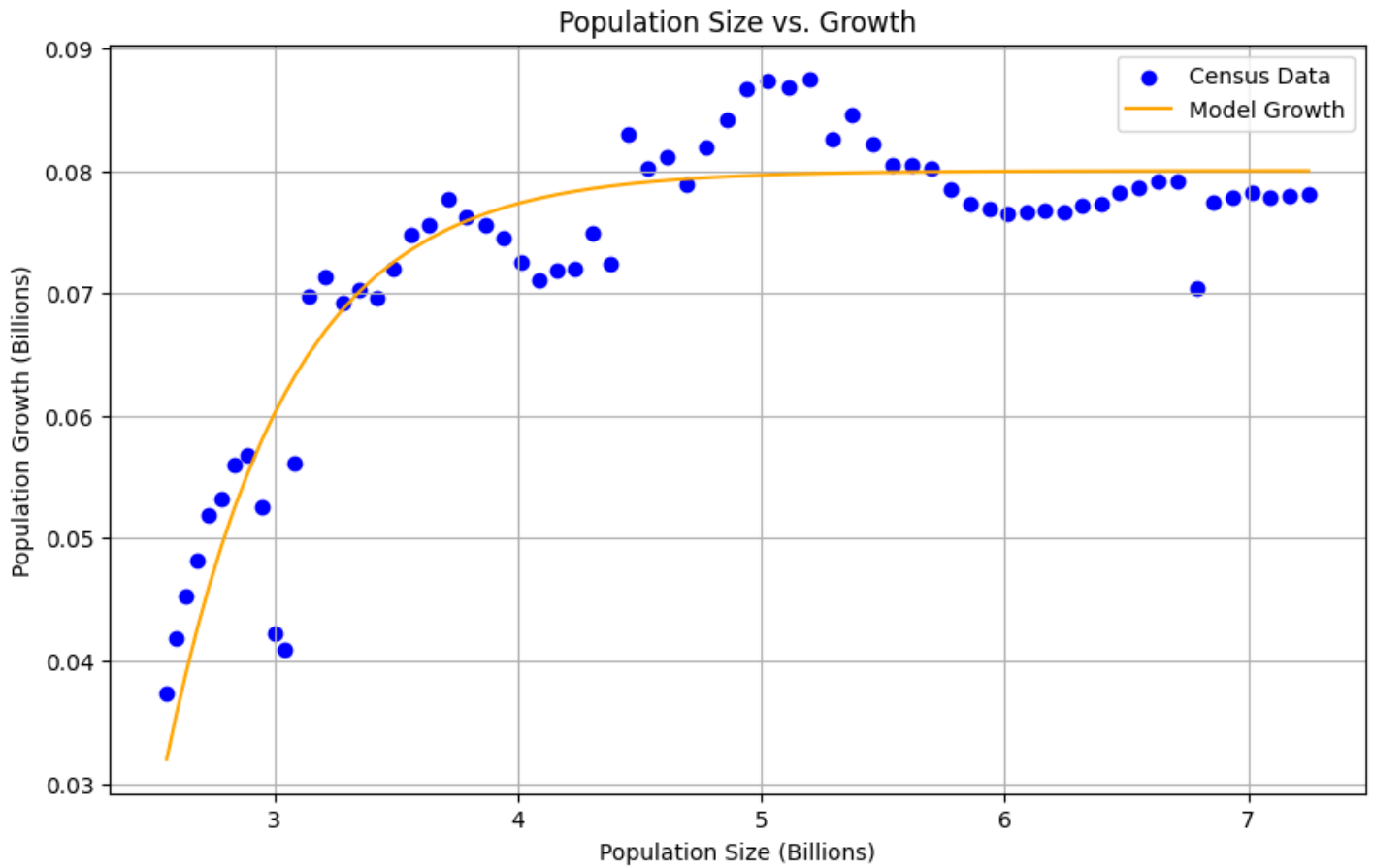
Task

- Try to extract just the growth data and plot it.

```
# Calculating annual growth
annual_growth = census.diff() # Using diff() to calculate the change in population each year


annual_growth = annual_growth.shift(-1)


# Plotting Population vs. Growth
plt.figure(figsize=(10, 6))
plt.scatter(census, annual_growth)
plt.xlabel('Population Size (Billions)')
plt.ylabel('Population Growth (Billions')
plt.title('Population Size vs. Growth')
plt.grid(True)
plt.show()
```


Population Size vs. Growth
Population Growth (Billions
0.09
0.08
0.07
0.06
0.05
0.04
3
4
5
6
7
Population Size (Billions)

```
annual_growth = census.diff().shift(-1) # Extract annual growth from census data

# Model Parameters
a, b, c = 0.08, -8, 2 # Coefficients for the model

# Initialize an empty list to store model growth data
growth_data = []

# Loop over the empirical population points to calculate growth
for population in census[:-1]: # Exclude the last year for matching dimensions
    growth = a + b / np.exp(c * population) # Calculate growth based on the model
    growth_data.append(growth) # Store the growth value
```


Population Size vs. Growth
Census Data
Model Growth
Population Growth (Billions)
Population Size (Billions)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
3
4
5
6
7

- Compute Errors for growth data
  - Compute mean absolute error
  - Compute maximum absolute error
  - Compute relative error

Laboratory Exercise (Blended Learning)

- What other functions can produce similar shapes
- Try other suitable functions and compute the error for each one.
  - mean absolute error
  - maximum absolute error
  - relative error

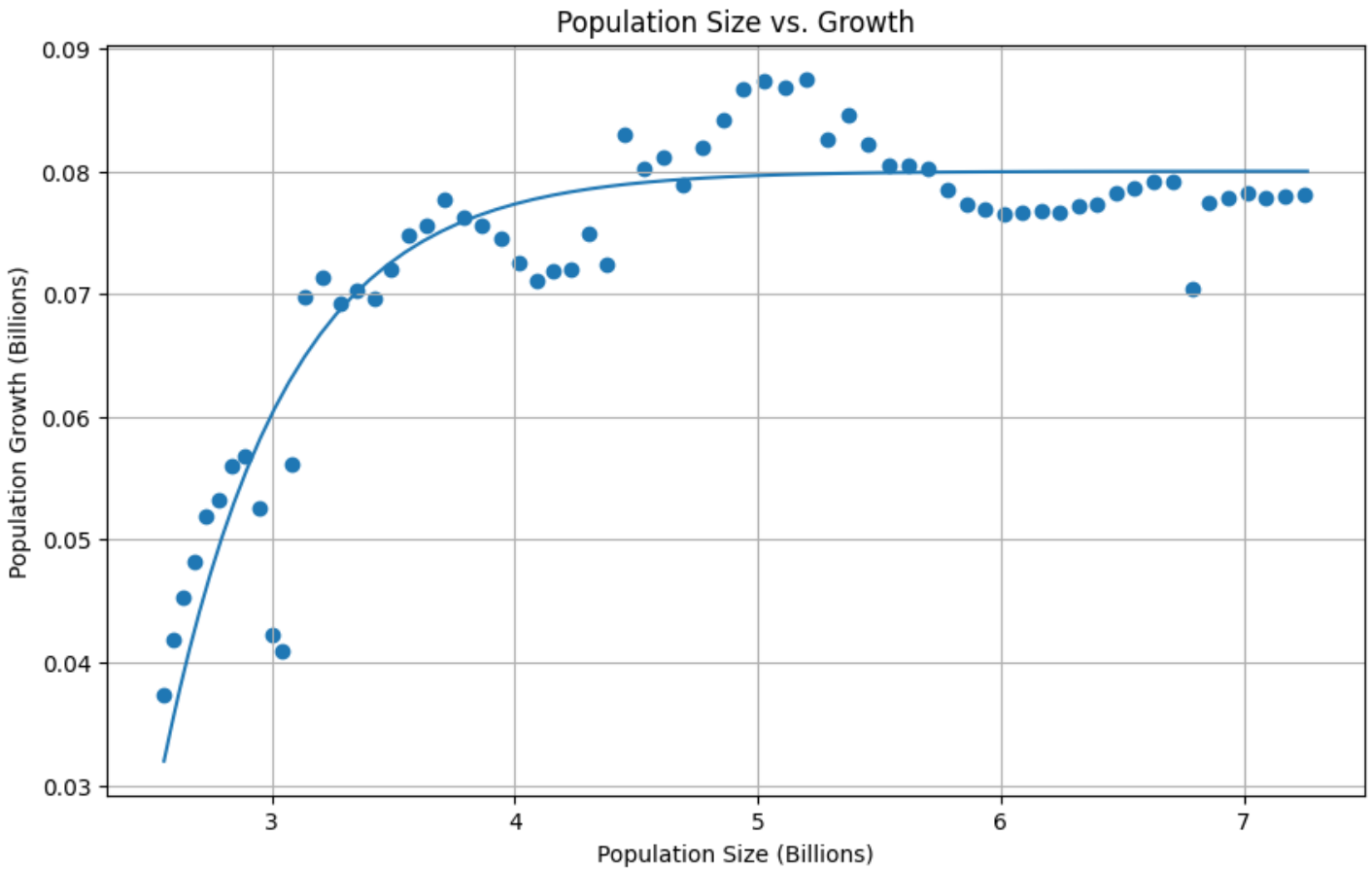
- Analise each function by commenting on its performance
- Incorporate the best function in the system model class

```python
class PopulationEstimateSystem:
    def __init__(self, p_0, a, b, c):
        self.p_0 = p_0 # Initial population
        self.a = a # Linear growth rate coefficient
        self.b = b # Quadratic growth rate coefficient
        self.c = c
        self.population = p_0 # Current population, starts at the initial value
        self.growth_history = [] # Store growth history

    def update(self):
        growth = self.a + self.b / (np.exp(self.c*self.population))
        self.population = self.population + growth
        self.growth_history.append(growth) # Store this update's growth
        return self.population
```

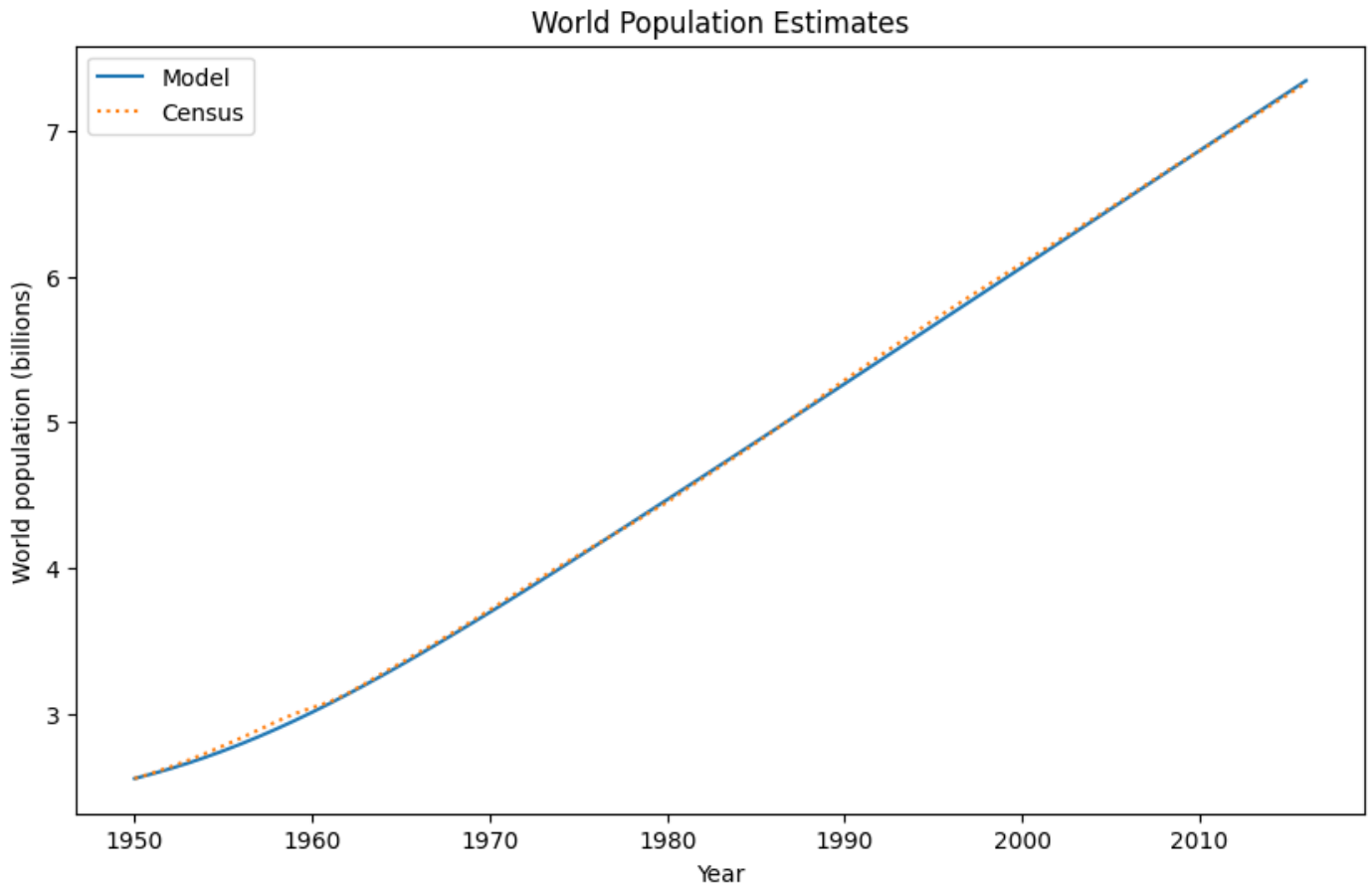
Continue The Class

```
t_0 = census.index[0]

t_end = census.index[-1]

#t_end = 2100

p_0 = census[t_0]

a = 0.08

b = -8

c = 2
# Create a PopulationEstimateSystem instance

population_system = PopulationEstimateSystem(p_0, a, b, c)
population = pd.Series([p_0], index=[t_0]) # initialize the population Series

population[t_0] = population_system.population # Initialize with the initial population

growth_data = [] # List to store growth values
for t in range(t_0, t_end):
        population[t+1] = population_system.update()
        growth_data.append(population_system.growth_history[-1])



# Calculating annual growth

annual_growth = census.diff() # Using diff() to calculate the change in population each year

annual_growth = annual_growth.shift(-1)
```

```
# Plotting Population vs. Growth
plt.figure(figsize=(10, 6))
plt.scatter(census, annual_growth)
plt.plot(population[:-1], growth_data,
  label='Model')
plt.xlabel('Population Size (Billions)')
plt.ylabel('Population Growth (Billions)')
plt.title('Population Size vs. Growth')
plt.grid(True)
plt.show()
```


Population Size vs. Growth
Population Growth (Billions)
Population Size (Billions)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
3
4
5
6
7

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(population.index, population,
 label='Model')
plt.plot(census.index, census, label='Census',
 linestyle=':')
plt.xlabel('Year')
plt.ylabel('World population (billions)')
plt.title('World Population Estimates')
plt.legend()
plt.show()
```


World Population Estimates
Model
Census
World population (billions)
3
4
5
6
7
1950
1960
1970
1980
1990
2000
2010
Year