Header:

Author: Crissy Hendrickson

Title: Bacteria Growth in Glucose

Group Members: Tori Steinberg, Cooper Linn, James Park

Analysis of the Problem:

When bacteria are first introduced into a new environment, there will be a certain lag time before the population starts to change. However, after the lag time has passed, the bacteria growth can be modeled by the Monod law which gives the growth rate as a function of nutrient concentration. Additionally, the bacteria death can be modeled by a first order rate so the overall rate of change of the bacteria population can be modeled by both its growth and death. Because the bacteria consume the nutrients to survive, the rate of change of the nutrient concentration can be modeled by the bacteria growth rate and the amount of nutrients that must be metabolized to make one bacterium per milliliter. If data is collected for the growth rate of bacteria in glucose over time, the parameters of the model can be fit to the data using Python. When the log of the bacteria concentration is used as y values, one can apply linear regression to different parts of the graph to get manually fitted parameter values. Additionally, one can solve the differential equations describing the rate of change of bacteria population and glucose concentration using odeint from SciPy. These manually fitted parameter values can then be used along with the solved differential equations as inputs in curve_fit to find the parameters which fit the model. Furthermore, parameters can be fit to the data using polynomials of different degrees rather than the growth model. However, it can be expected that the growth model will produce more reasonable extrapolations of the number of bacteria for t=150 hours because when the polynomials are used the fit tends to approach infinity or negative infinity depending on the order as time increases.

Model Design:

Computational models are simplified representations of phenomena, systems, or processes. They are used to help us study and understand different aspects of life and can lead to predictions involving responses in un-tested scenarios. Often because of certain assumptions and simplifications, they are much easier to work with than the systems they represent and can point to new insight and comprehension. To translate a problem into a computational model there are five main steps: analysis of a problem, model development and formulation, model implementation, model verification, and model interpretation. The first step, analysis of a problem, involves determining what the question and unknowns are and what the answer would look like. The second step, model development and formulation, includes gathering relevant data, making assumptions, determining variable relationships and units, and writing down specific equations or rules. Model implementation involves writing a program in a computer language to solve the problem while model verification involves testing special cases whose answers are known to confirm the solution is accurate and the program does what it is supposed to do. Finally, model interpretation includes answering the main question sometimes with figures or tables and discussing more about the solution: if it was expected, what it means and how it could change if the assumptions are relaxed.

In the model for Bacteria Growth in glucose, several assumptions were made before the model was implemented. The model was assumed to be dynamic since it depends on time and deterministic since there is no element of chance and the solution is expected to be the same under the same conditions. It is also assumed to be a point model since the solution does not depend on the location in space. Because the individual bacteria will be dying and multiplying at different times and taking up all available space along with the glucose, time and space are assumed to be continuous in the model. Furthermore, the number of bacteria and the amount of glucose is assumed to be continuous because the amount of each thing can include all points on the number line. In the model, the initial amount of glucose was also given to be 0.2 microgram/ml while the initial number of bacteria was given to be 50 bacteria/ml. The model was developed over 100 hours with one hour between each data point so the trend on the graph could be determined and one could see if the model fit the collected data.

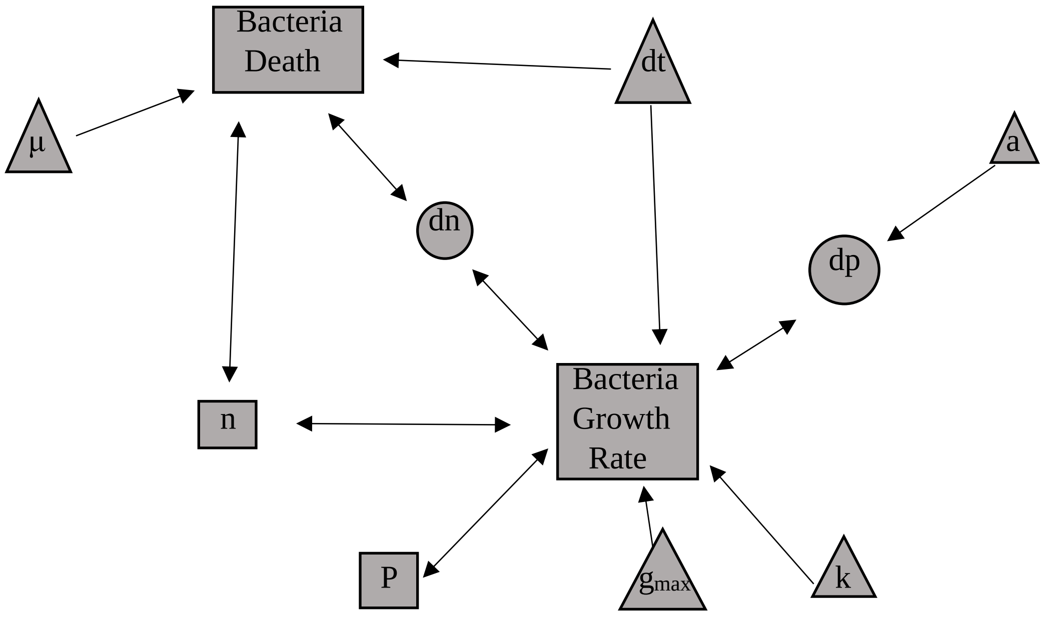
In the model development, several variables and equations were used to help find the solution. The maximum growth rate of the bacteria, g_{max} bacteria/ml(hour), the Monod constant, k microgram/ml, and the death rate of the bacteria, μ hour^{-1}, are all constants. Additional constants include the change in time, dt hours, and the amount of nutrients that must be metabolized to make one bacterium per milliliter, a CFU/(mg/ml). The number of bacteria bacteria/ml, is a container variable as is the amount of glucose microgram/ml. Furthermore, the change in number of bacteria bacteria/ml and the change in the amount of glucose microgram/ml are dependent variables. Using the previously described variables, a system of differential equations can be used to represent the rate of change of the number of bacteria and amount of glucose over time. The first equation

$$\frac{dn}{dt} = \frac{npgmax}{p+k} - \mu n \qquad (1)$$

describes the rate of change of the bacteria population where $\frac{npgmax}{p+k}$ is the growth rate of the bacteria and μn is the death of the bacteria. The second equation

$$\frac{dp}{dt} = \frac{npgmax}{a(p+k)} \qquad (2)$$

describes the rate of change of the amount of glucose where $\frac{npgmax}{p+k}$ is again the growth rate of the bacteria. The two equations can then be solved numerically in Python to give the amount of bacteria and glucose. Overall, the variables and equations are represented by the following diagram.

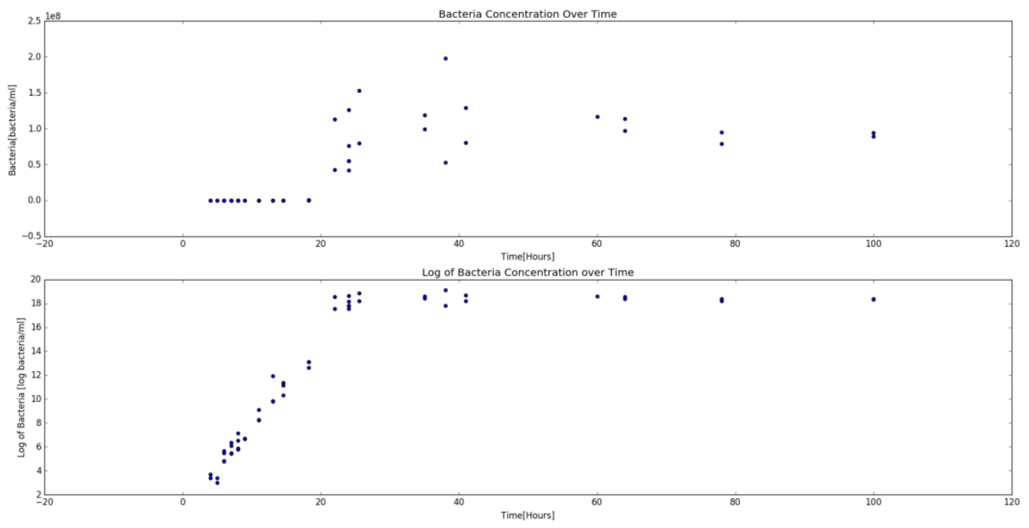


Model Solution:

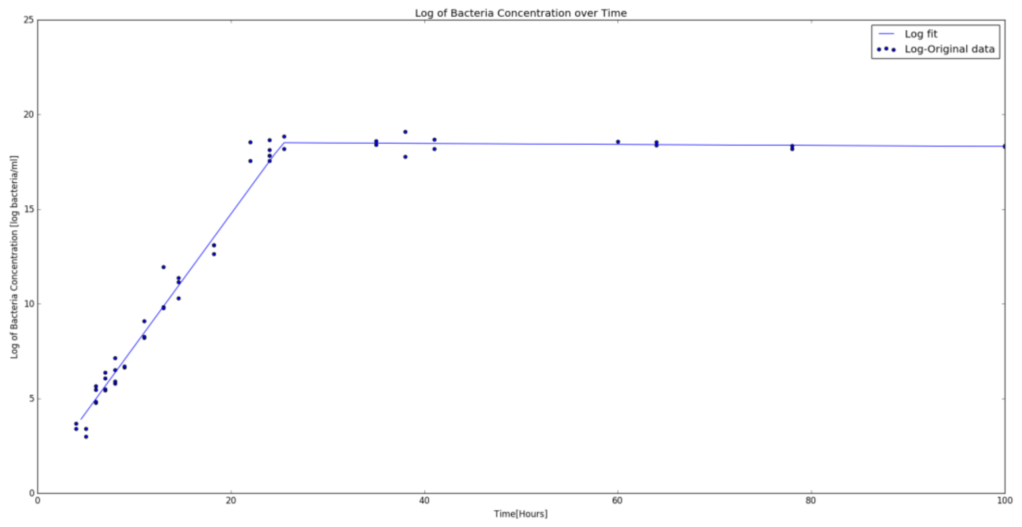
To find the solution, the code at the end of the report was run in python. First the data was imported from the web using urllib and split up into an array of time values and an array of corresponding bacteria concentration. A graph was then made of the bacteria concentration over time so the data could be visualized. Additionally, a graph was made of the log of the bacteria concentration over time so it would be easier to fit the parameters to the model. Then manually fitted parameters were found for k, a, μ, and g_{max} using linear regression so they could later be used as inputs into curve_fit. Specifically, a line was fit to the last 11 data points and then the slope of that line was used as a temporary value for μ. Then a line was fit to the first 29 data points so a temporary value of g_{max} could be found by adding its slope to the temporary value of μ. Additionally, the temporary value for a was found by multiplying the temporary value for μ with the maximum bacteria concentration and dividing by the corresponding glucose concentration. A temporary value for k was arbitrarily chosen to be a small number. After the parameters were manually fitted, methods were written using odeint from Scipy to solve Eq.(1) and (2) numerically. It should be pointed out that the log was taken of the odeint so the units would be consistent. Then the curve_fit function was used with the manually fitted parameters and the numerically solved equations as inputs so the parameters could be fit to the model. Finally, a graph was made which had the bacteria data and a line with the fitted parameters. For comparison, a graph was also made with a fit that used polynomials so the two methods could be compared. To verify the solution, graphs were made with variations to some of the fitted parameters which came from the curve_fit function. They were then compared with the original plot to see if the new plots behaved as expected.

<u>Results, Verification, and Conclusion:</u>

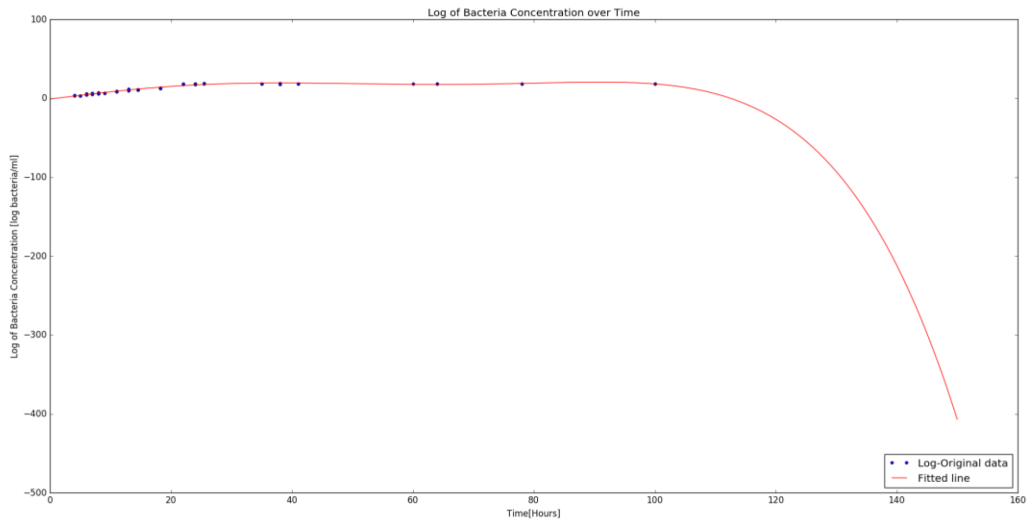
The following graphs show the bacteria concentration over time and the log of the bacteria concentration over time for the provided data.



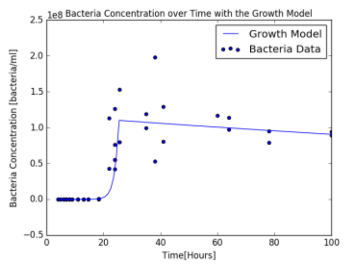
On the second graph, linear regression was used to help find manually fitted parameter values for k, a, μ, and g_{max} to be .007 microgram/ml, 89489425.86 CFU/(mg/ml), .00238 hour^{-1}, and .770 bacteria/ml(hour). Once the manually fitted parameter values were determined, they could be used as inputs into curve_fit along with the numerically solved Eq.(1) and (2). Using curve_fit, the parameters of the model were fit to the data in the graph below.

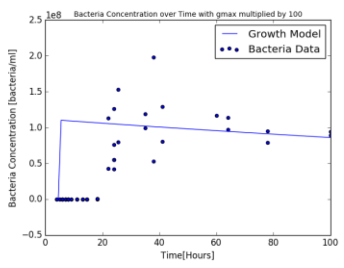


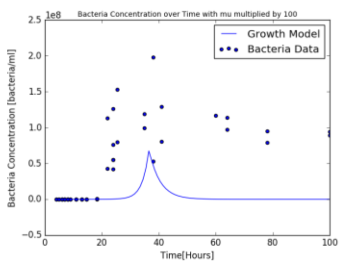
The parameter values of k, a, μ, and g_{max} were found to be 8.07x10^{-8} microgram/ml, 5.51x10^{8} CFU/(mg/ml), .0026 $hour^{-1}$, and .702 bacteria/ml(hour) which all make sense within the context of the problem. As one can see on the graph, the bacteria concentration initially increases until the glucose runs out and the concentration levels off. Next time to improve results, more data might help the line be a better fit a better fit so there is less error. The data can also be fit using polynomial of different degrees as shown in the graph below.

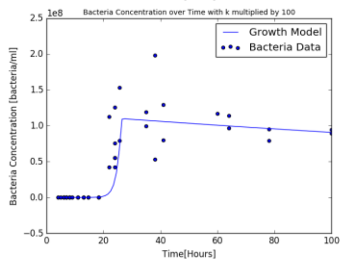


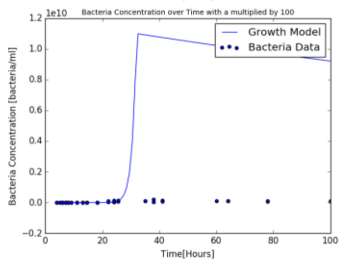
As one can see, the growth model produces more reasonable extrapolations of the measured CFU for t=150 hours because when the polynomials are used the fit approaches negative infinity. To verify the solution, plots were created with different parameter values and compared with the original plot to see if the system behaved as expected.











When g_{max} is increased by 100, one can expect the bacteria concentration to grow very quickly at first before it evens out. If μ were increased by 100, one could expect the bacteria concentration will approach zero over time. Additionally, if k were increased by 100, one could expect the growth rate to decrease. Finally, if a were multiplied by 100 one could expect the maximum value of the bacteria concentration to increase by a factor of 100. Since all the plots match what is expected, the solution is verified.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import urllib
from scipy.integrate import odeint
from scipy.optimize import curve_fit

#imports data
bacteria_data=
urllib.request.urlopen("http://nemenmanlab.org/~ilya/images/d/d8/Bacteria.txt")
data_set=np.loadtxt(bacteria_data, delimiter=',')
data_set_sorted = data_set[data_set[:, 0].argsort()]

#splits up the data into an array of time values and an
#array of cooresponding bacteria concentration
N=np.size(data_set)
Time=data_set_sorted[0:N:1,0]
Bacteria_Concentration=data_set_sorted[0:N:1,1]
tf = 150

#graphs the bacteria concentration over time
plt.subplot(2,1,1)
plt.scatter(Time,Bacteria_Concentration)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria[bacteria/ml]')
plt.title('Bacteria Concentration Over Time')
plt.show()
#graphs the log of the bacteria concentration over time
plt.subplot(2,1,2)
```

```
t_data = np.transpose(data_set_sorted)[0]
n_data = np.log(np.transpose(data_set_sorted)[1])
n_original = np.transpose(data_set_sorted)[1]
t = np.linspace(0, tf, tf +1).flatten()
plt.scatter(Time, n_data)
plt.xlabel('Time[Hours]')
plt.ylabel('Log of Bacteria [log bacteria/ml]')
plt.title('Log of Bacteria Concentration over Time')
plt.figure()

#initial glucose and bacteria concentrations
init = [0.2, 50]

# k = Monod constant = p*10^-100
k_guess = 0.007

# mu = bacteria death rate
B = t_data[41:52]
B = np.vstack([np.ones(B.size), B]).T
y_intercept, slope = np.linalg.lstsq(B, n_data[41:52])[0]
mu_guess= -slope

# gmax = maximum bacteria growth rate
C = t_data[0:29]
C = C.reshape((C.size, 1))
z = n_data[0:29]
z = z.reshape(z.size, 1)
slope2 = float(np.linalg.lstsq(C, z)[0])
gmax_guess = slope2 + mu_guess

# a = CFU/mg*ml

a_guess = (mu_guess * Bacteria_Concentration[41] / (0.2/38))

timearray = np.arange(0, 52, 1).flatten()
# Monod law = bacteria growth rate
def diff_eq(values, timearray, gmax, mu, k, a):
    p = values[0]
    n = values[1]
            # n = bacteria concentration
    dn_dt = n * gmax * p / (p+k) - mu*n
            # p = glucose concentration
    dp_dt = - n * gmax * p / (p+k) * (1/a)
    return dp_dt, dn_dt

def odeint_diff_eq(t, gmax, mu, k, a):
    return odeint(diff_eq, init, timearray, args = (gmax, mu, k, a))[0:N:1,1]

def log_odeint_diff_eq(timearray, gmax, mu, k, a):
    return np.log(odeint(diff_eq, init, timearray, args = (gmax, mu, k,
a)))[0:N:1,1]

gmax1, mu1, k1, a1 = curve_fit(log_odeint_diff_eq, t_data, n_data, p0 =
(gmax_guess, mu_guess, k_guess, a_guess), bounds=([0,0,0,0], [1, 1, 1,
10000000000]))[0]
```

```
#equations redefined to fit a larger time period
timearray2 = np.arange(4.5, 108.5, 1).flatten()
def diff_eq2(values, t, gmax, mu, k, a):
    p = values[0]
    n = values[1]
            # n = bacteria concentration
    dn_dt = n * gmax * p / (p+k) - mu*n
            # p = glucose concentration
    dp_dt = - n * gmax * p / (p+k) * (1/a)
    return dp_dt, dn_dt

def odeint_diff_eq2(t, gmax, mu, k, a):
    return odeint(diff_eq, init, timearray2, args = (gmax, mu, k, a))[0:N:1,1]

def log_odeint_diff_eq2(t, gmax, mu, k, a):
    return np.log(odeint(diff_eq, init, timearray2, args = (gmax, mu, k,
a)))[0:N:1,1]

#graph odeint with parameter values from curve fit for bacteria vs. time graph
plt.plot(timearray2, log_odeint_diff_eq2(t, gmax1, mu1, k1, a1), label='Log fit')
plt.xlim([0, 100])
plt.title('Log of Bacteria Concentration over Time')
plt.xlabel('Time[Hours]')
plt.ylabel('Log of Bacteria Concentration [log bacteria/ml]')
plt.scatter(Time, n_data, label ='Log-Original data')
plt.legend()
plt.show()
plt.figure()

#polynomial fit
plt.plot(t_data, n_data, 'o', label='Log-Original data', markersize=4)
A = np.vstack([np.ones(len(t_data)), t_data, t_data**2, t_data**3, t_data**4,
t_data**5]).T
a, b, c, d, e, f = np.linalg.lstsq(A, n_data)[0]
plt.plot(t, a+b*t+c*t**2+d*t**3+e*t**4+f*t**5, 'r', label='Fitted line')
plt.legend(loc=4)
plt.title('Log of Bacteria Concentration over Time')
plt.xlabel('Time[Hours]')
plt.ylabel('Log of Bacteria Concentration [log bacteria/ml]')
plt.show()
plt.figure()

#model verification
plt.subplot(2,3,1)
plt.plot(timearray2, odeint_diff_eq2(t, gmax1, mu1, k1, a1), label='Growth Model')
plt.xlim([0, 100])
plt.title('Bacteria Concentration over Time with the Growth Model', size=12)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria Concentration [bacteria/ml]')
plt.scatter(Time, Bacteria_Concentration, label = "Bacteria Data")
plt.legend()
plt.show()
#gmax
plt.subplot(2,3,2)
plt.plot(timearray2, odeint_diff_eq2(t, 100*gmax1, mu1, k1, a1), label='Growth
Model')
plt.xlim([0, 100])
plt.title('Bacteria Concentration over Time with gmax multiplied by 100', size=10)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria Concentration [bacteria/ml]')
plt.scatter(Time, Bacteria_Concentration, label = "Bacteria Data")
plt.legend()
plt.show()
#mu
plt.subplot(2,3,3)
plt.plot(timearray2, odeint_diff_eq2(t, gmax1, 100*mu1, k1, a1), label='Growth
Model')
plt.xlim([0, 100])
plt.title('Bacteria Concentration over Time with mu multiplied by 100', size=10)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria Concentration [bacteria/ml]')
plt.scatter(Time, Bacteria_Concentration, label = "Bacteria Data")
plt.legend()
plt.show()
#k
plt.subplot(2,3,4)
plt.plot(timearray2, odeint_diff_eq2(t, gmax1, mu1, 100000*k1, a1), label='Growth
Model')
plt.xlim([0, 100])
plt.title('Bacteria Concentration over Time with k multiplied by 100', size=10)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria Concentration [bacteria/ml]')
plt.scatter(Time, Bacteria_Concentration, label = "Bacteria Data")
plt.legend()
plt.show()
#a
plt.subplot(2,3,5)
plt.plot(timearray2, odeint_diff_eq2(t, gmax1, mu1, k1,100*a1), label='Growth
Model')
plt.xlim([0, 100])
plt.title('Bacteria Concentration over Time with a multiplied by 100', size=10)
plt.xlabel('Time[Hours]')
plt.ylabel('Bacteria Concentration [bacteria/ml]')
plt.scatter(Time, Bacteria_Concentration, label = "Bacteria Data")
plt.legend()
plt.show()
```