# School of Physics and Astronomy


## Senior Honours Project
## Physics 4

# How Bacteria Form 3D Structures

**Yusong Tian**
**24th April 2018**

**Abstract**

In this project analysis are done with code written in python on experimental data acquired in previous research, aiming at understanding the nature of how bacteria form 3d structure. It was found that the bacteria grows into a layered structure on agarose and the process can be described with the model of coupled differential equation. The speed of growth is related to the concentration of agarose.

**Declaration**

I declare that this project and report is my own work.

Signature: Yusong Tian Date: 24/ 04/2018

**Supervisor:** Prof. Rosalind Allen, Dr.Nikola Ojkic 10 Weeks

# Contents

# 1 Introduction

How biological cells form complex spacial structures is one of the most fundamental question in biological physics. In this project the spacial organisation of surface growing *E.coli* (*Escherichia coli*) colonies on agarose was studied, aiming at understanding the 3d spacial structure of this bacteria that is generally used everyday in microbiological labs.[1] Studies on the growth of bacteria on soft matter may bring about understanding of its growth, laying foundation for future research like bacteria growth in tissue that may lead to application in the future.

This project is a continue of previous study [3]. In the previous study experiments were done, videos of *E.coli* grown on different agarose concentration and different nutrient were obtained. It was found that there is a 2d to 3d transition in the growth and a layer structure which were named "wedding cake" because of its 3d shape. This layered structure was observed in other researches as well [4]. In this experiment two or three layers were observed in each colonies.

Then further procedures were done on the videos, by which the radius of layers versus time were acquired and stored as data files in .txt format.

There were 34 data files and in this project the data were analysed with help of code written in Python and an mathematical model was tested.

It was found that the model of coupled differential equations is a good description of this layered structure and the growing process. The area of the first layer grows exponentially (which means a linear line when data being shown in logarithmic scale) and upon showing up of the second layer, the growth slows down. For those replicates that has three layers, the increase of the area of the second layer slows down as well when the third layer shows up.

Summing up the area of all layers gives a line that is linear at the beginning (last longer than the linear part of the first layer), which indicates that the assumption of the total quantity of the bacteria growing exponentially is possible to be correct. Although the sum no longer form a straight line under logarithmic scale in the end, which may indicate deeper structures which cannot be observed with the method used in previous research.

The gradient of the linear growing part of the first layer was obtained, a plot gradient versus agarose concentration shows that the gradient generally decrease with the increase of agarose concentration, with the exception of the gradient of the fast growing group growing on 4% agarose is slightly larger than that on 3%. Which indicates more complicated relation. The gradients of the increase of the area shows how quickly the colony expands, which means the speed of the reproduction of the bacteria. This result shows that bacteria grow slower on higher concentration of agarose.

# 2 Background [3]

This project is a continue of the experiment done by Dr. Lloyds and the relevant work is included in mainly chapter 8 in his PhD thesis.

In previous research, experiments were carried out and some movies of bacterial colonies growing on agarose (a type of polymer extracted from seaweed, generally used as the substance to grow bacteria on) were acquired under the microscope. The type of bacteria was *E.coli*, which is widely used in researches as model organism whose properties are representative as general properties of bacteria. The colonies were grown under different condition. The varied conditions are:

1. 2 different nutrient condition, resulting to the bacteria to grow with two different speed, which is referred to as fast and slow.

2. 4 different agarose concentration for the fast group: 1.5%, 2%, 3% and 4% and 3 different agarose concentrations for the slow group: 2%, 3.5%, 4%. These numbers are not chosen for particular reason but convention and the uncertainty result from the complicated procedure of making agarose.

There are some repetitions for each condition, the number of repetitions depend on the number of unbroken colonies. In this project they are labelled 1, 2, 3, etc. for distinguishing purpose.

The experimental result shows that bacterial colonies grow in a layered structure which was named the"wedding cake" structure.

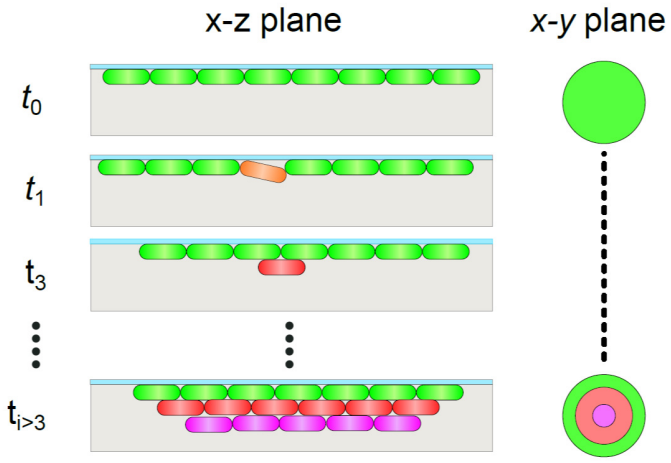
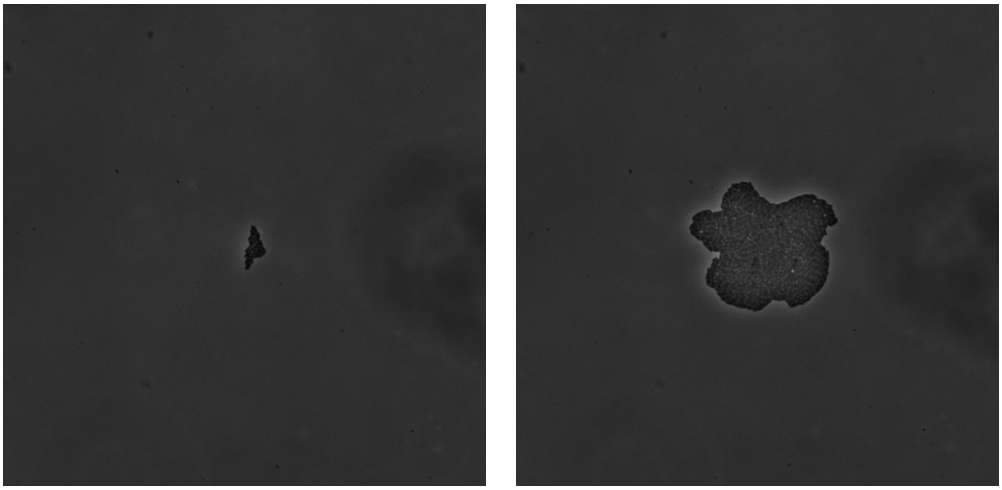



Figure 1: The wedding cake structure. At first the bacteria grow in a layer, after a certain time a second layer shows up, and then the third. (Image from [3])

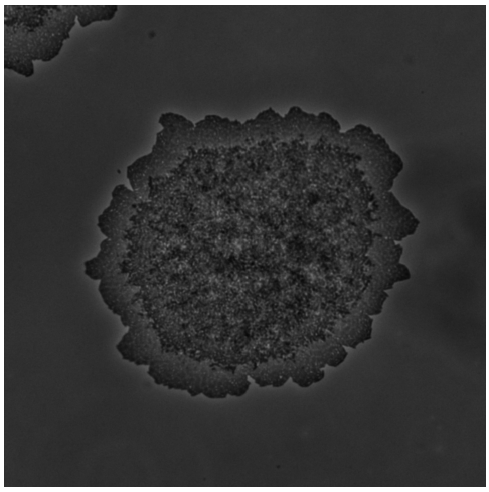
Figure 2 shows an example of some frames from one of the videos. Both the videos and the frames in the videos were stored with format .tif and can be viewed and edited with software "Fiji" (also called "ImageJ").



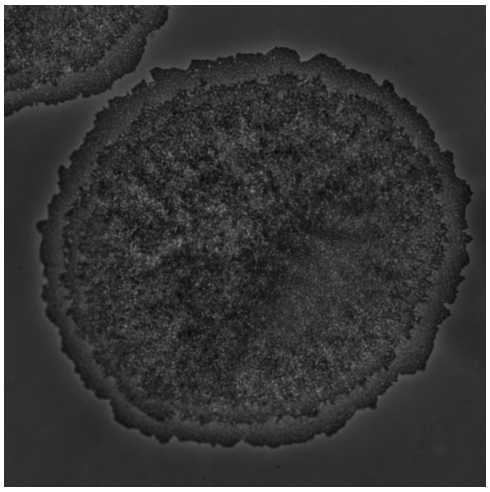
(a) Frame 1



(b) Frame 33



(c) Frame 65



(d) Frame 97

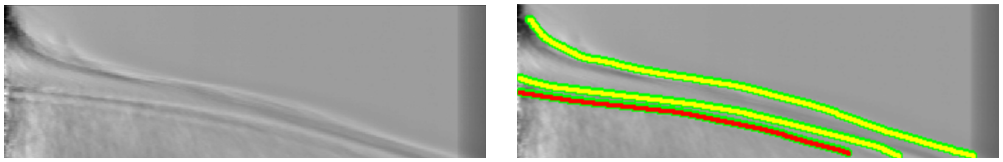
Figure 2: Example of frames from videos. This is a fast growing one in 2% agarose, the first repetition. The sign of second layer showing up can be seen in frame 33 (the little black spots in the colony) and later on it becomes clear that there are different layers in the colony. The edge between the first and second layer is obvious while the third layer is more difficult to be observed.

Then analysis on these videos was carried out. As the colonies are not perfect circles, every frame in the videos was sliced along several radii, giving rows of single pixels, each having the contrast (which were given by Fiji) of the slice along one radius of the colony. Then the contrast of these slices were averaged, giving a final row of pixels each carrying the averaged contrast of the corresponding radius of the colony.

Then the rows of pixels from each frame are stacked together, generating a figure that can be analysed quantitatively. As there is a difference of contrast at the edge of each layer, tracking the place when the contrast is particularly different can give the change of the radius of each layers versus time.

With plug-in "JFilament", the lines on the generated figure with particularly different contrast were found out. (See figure 3.) The lines (as well as the data lines for different layers in the figure show in this report layer on) were named "snakes" and the data for

them are saved in a .txt format. In the data files the data stored are the position of dots that forms the snakes. Taking the origin point as the top left point of the figure, the vertical axis pointing downwards represent time and the horizontal axis pointing right represents distance (radius). The units are pixels. In the original video images are taken every 3 minutes and pixel size for the distance is 0.06449 micron/pixel, by these data the time in hour and radius in micron can be calculated.



(a) The generated image (b) Added snakes

Figure 3: An example of the generated image and snakes. In these images each row of pixels has a contrast that is the average of the radii in one frame from the video. Thus the vertical axis represents time: lower rows represent later radii. Horizontal axis represent the distance. In figure (b) the lines that represents the boundary of the different layers are marked by the "snakes". For example the one marked red is the third snake, which shows up last and are not visible in some of the other images generated from experiments, meaning that some of the data files contain only data for two snakes. This example is the fast growing colony in 2% agarose, the first repetition.

These were done by previous researchers in our group. Within this project the acquired data was analysed, aiming at testing out a mathematical model that lead to better understanding of this bacteria growing process.

# 3 Theory [2]

Bacteria reproduce by division and the population increase exponentially.

$$N(t) = N(0)e^{\lambda t} \tag{1}$$

where: $N(t)$ is the population of the bacteria at time t;
$N(0)$ is the number of bacteria at the starting time;
$\lambda$ is the parameter that relates to the speed of growth.
Taking log on both side:

$$log(N(t)) = log(N(0)) + \lambda\, t \tag{2}$$

Which means if the number of bacteria is plotted in logarithmic scale, the plot would approximately be a straight line.

Because of the resolution of the images and difficulty counting the exact number, the quantity of bacteria is rather difficult to get. While as the measured quantity is the radius of the colony, the area of the colony would be easily calculated. Assuming that the number of bacteria is evenly distributed, the area is a good representation of the number of bacteria.

Differentiating equation 2 we have the differential equation:

$$\frac{dN}{dt} = \lambda N \tag{3}$$

In the context of bacteria growing in a layered structure, taking the transition to a second layer and third layer in to account, assuming that the transition from first layer to the second layer is $k_1(t)$, the transition from the second layer to the third is $k_2(t)$, the growth of bacteria colonies follows the coupled differential equations:

When there is only one layer:

$$\frac{dN_1}{dt} = \lambda N_1 \tag{4}$$

When the second layer shows up:

$$\frac{dN_1}{dt} = \lambda N_1 - k_1(t) \tag{5}$$

$$\frac{dN_2}{dt} = \lambda N_2 + k_1(t) \tag{6}$$

After the third layer shows up:

$$\frac{dN_1}{dt} = \lambda N_1 - k_1(t) \tag{7}$$

$$\frac{dN_2}{dt} = \lambda N_2 + k_1(t) - k_2(t) \tag{8}$$

$$\frac{dN_3}{dt} = \lambda N_3 + k_2(t) \tag{9}$$

Overall:

$$N_1 + N_2 + N_3 = N_{total} \tag{10}$$

$$\frac{dN_{total}}{dt} = \lambda N_{total} \tag{11}$$

where:

$N_{total}$ is the total population of bacteria in the colony and $N_1$, $N_2$, $N_3$ are the population of bacteria in the first, second and third layer respectively.

It is assumed that the overall quantity of bacteria still grows exponentially although forming a layered structure.

# 4 Method

A code (listed in the appendix) with Python is written to process the data from the files. In order to compare the result from different parameters (fast or slow growing, agarose concentration), the .txt data files were named in a systematical way, with format:

$$*_1\_snakes\_*_2\_*_3\ pc\_*_4 \tag{12}$$

where:

$*_1$ take number from 1 to 34, being the total index of the files; $*_2$ is letter "f" or "s", represent the file is the data for fast or slow growing colonies; $*_3$ take the value for agar concentrations; $*_4$ is index of the replicates.

The data were processed and plotted, by observing the figure, the nature of bacteria growth can be observed and mathematical model can be tested. For example if the area of bacteria is linear with time under logarithm scale, then the assumption that bacteria grow exponentially can be confirmed. It can also be found out how different parameters influence the growth.

The code:

Opens the data files. Each data file contain data for 2 or 3 snakes. A function was written to separate the data in one file for different snakes and store them in the form of lists. Each item in the list contain a list of data for one snake. As mentioned in previous sections, both the data for time and radius are stored with unit "pixels". Within the function the corresponding time is calculated in hour and radius in micron, then from the radius the area of the colony is calculated assuming the shape is a circle. Then time is stored in a list, area as well as which snake this is, the parameters used in this specific experiment and the index of this replicate are stored in another list. For doing the least squares fit, the code log the area and store this number in the list as well later on.

Another function was written in the code to do a least squares fit, calculate the gradient and intercept from the linear parts.

The data from experiments are plotted and processed to test the model. The code were divided into different parts to generate different type of figures:

1. The gradient and intercept of the linear part. Average and error bar are calculated and shown as well.

2. The area of different layers of the colonies versus time.

3. For the first snake this code can subtract the intercept obtained from the fitting and show the result, aiming at getting rid of the influence of the area that the colonies had already grown into when the video of the colony was taken. Although this did not generate any insight thus an analysis was not included in this report.

4. Summing up the area of all the layers obtained for each colony and show the result in figures.

# 5 Results

## 5.1 The Three Snakes

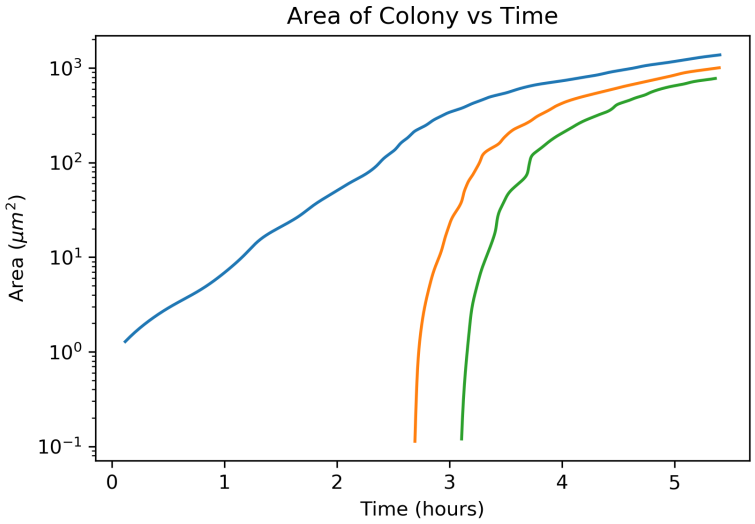



Figure 4: An example for how the area of the three layers change with time for one single colony (this example plot is generated from the data in the file "07_snakes_f_3pc_1"). The blue line is from the data from the first snake and describes the first layer of bacteria, the orange line is the second snake, represents the data for the second layer, and the green line is the third. It can be seen that there is indeed a linear relation of the area (which shows the number of bacteria) under logarithmic scale. Upon appearing of the second snake, the growth of the first layer becomes slower, and when the third snake shows up the growth of the second layer becomes slower as well.

From figure 4 it can be seen that there is indeed a linear part in the growth of all three layers. The gradient of the snakes represent how fast the area increase. The growth is in accordance with the assumption of the model: growing exponentially and then a transfer appears, the growth of the first layer slows down. And the second layer grows with a larger gradient as apart from the normal reproduction of the bacteria, there is a transition from the first layer as well, then the bacteria of the second layer start the transaction into the third layer.

The speed of growth of the third layer reduces as well eventually, which may indicate a possible fourth layer, or growth of other structure deeper than three layers showing up but not observable by the current measure: the data come from the contrast at the edge of the layer from the top view, the deeper the layer is, the more difficult it is to distinguish the edge thus difficult to be tracked. It can also be that the growth slows down because of the nutrient consumption.

For some condition or replicate there are only 2 snakes observable, which may be because the edge of the third snake is already too obscure to be distinguished, or the time isn't enough for deeper structure to show up.

## 5.2 Result for all Experiments

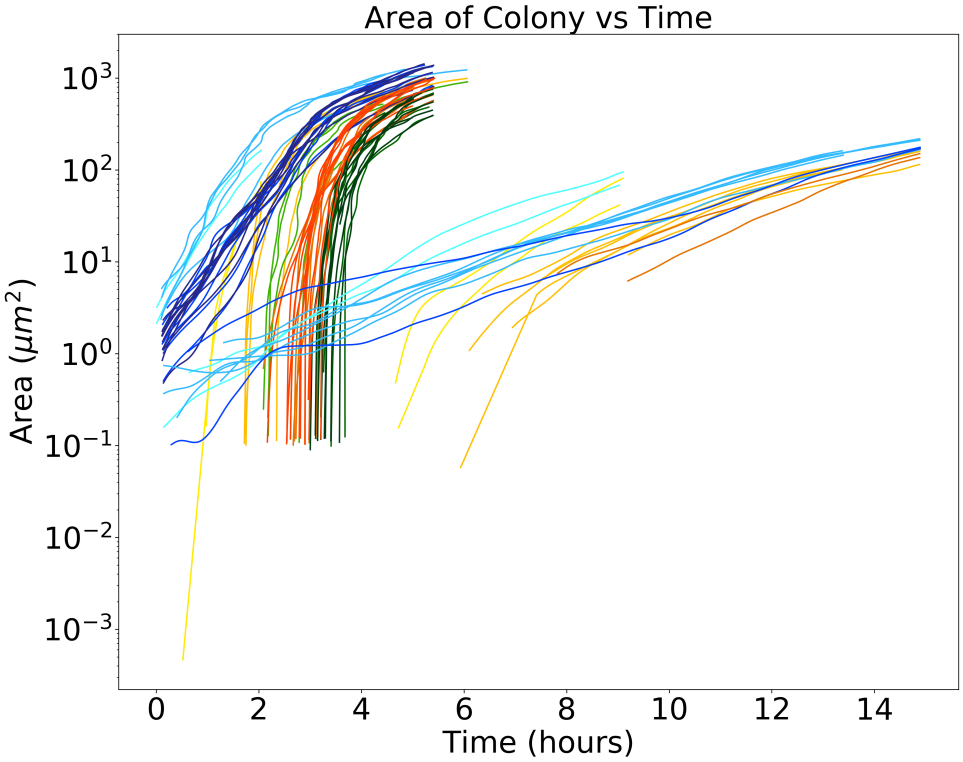



Figure 5: The area of the colony versus time for all experiments. The blue lines shows the data for the first snakes (first layer of the colony) for each experiment, the yellow or orange colour shows the second and green the third. Different shade are used to distinguish between different agar concentrations: lighter colour shows smaller concentration. The snakes that ends earlier are the fast growing ones and those last for longer are the slow ones.

Figure 5 shows that same trend can be observed for the data obtained from all experiments: growing linearly with time under logarithmic scale and slows down when the second snake shows up.

There are some difference in the position of the snakes. For the colonies grown in smaller agarose concentration, the second and third layer appears earlier than that for larger agar concentration.

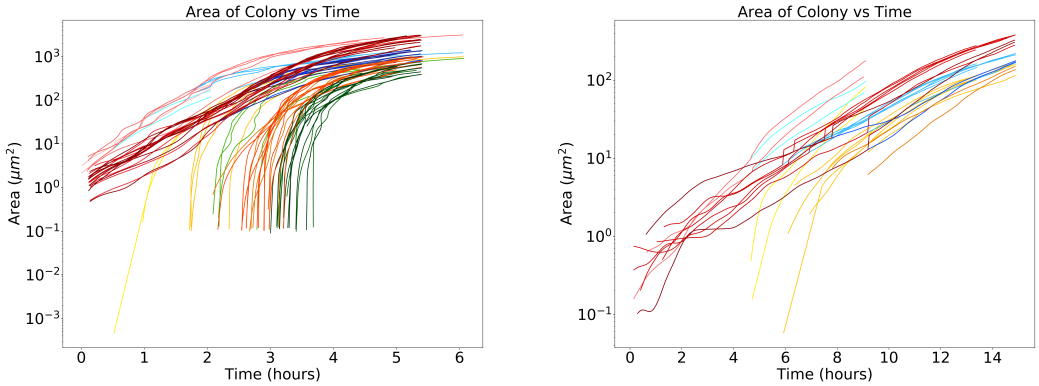
This can be explained with the elasticity of agar: bacteria get more resistance in agarose that has larger concentration, thus would take more time to invade deeper.

From the previous result of the gradient of the growth it can be seen that bacteria grow slower in higher agar concentration. It is reasonable to assume that the invasion would only happen provided that the previous layer has grown into a certain size, thus slower growth of the first layer lead to later appearance of the following layers.

Another thing to be taken into consideration is that in the figure, the position of the snakes are affected by how large has the colony already grown into when they start to be videoed. I.e. the intercept of the snakes with the y axis would show the area of the first snake when the video starts (at time 0). This does not reveal anything about the nature of bacteria growth, just showing that the starting points are different.

## 5.3 Total Quantity of Bacteria

For every data file, the area for all the snakes are summed up to show the total quantity of the bacteria.




(a) Fast growing

(b) Slow growing

Figure 6: The red lines shows the total area of all the layers. There are obvious cliffs on certain lines, which is because the area for some second or third layers is to difficult to be observed thus the early growth of those layers were not registered, creating a cliff when those layers are suddenly taken into account.

From the data for the total area of all layers, it can be seen that the linear part is indeed lengthened. This is more obvious in the slow growing group, for the fast growing ones, the lines still bend later on, the reason might be a possible fourth layer that wasn't observed, other structure such as random gathered bacteria without specific layer structure. From confocal florescence microscopy used in the research in previous stage [3] it can be seen that the growth of bacteria indeed goes deeper than simply three layers, agreeing with this assumption. While the possibility that other restriction such as consumption of nutrient playing a role is not excluded.

## 5.4 Gradient of the Linear Part of the First Layer

Least squares fit was done for the linear growing part of the first layer, which is the part before the second layer shows up, and the gradient and intercept were calculated for all files.

The gradient shows the speed of growth of the area of the colonies, so the data for the fast growing group of experiments has much higher gradients than the slow growing ones. Figure 7 illustrates that the data as well as their average and error indicate that

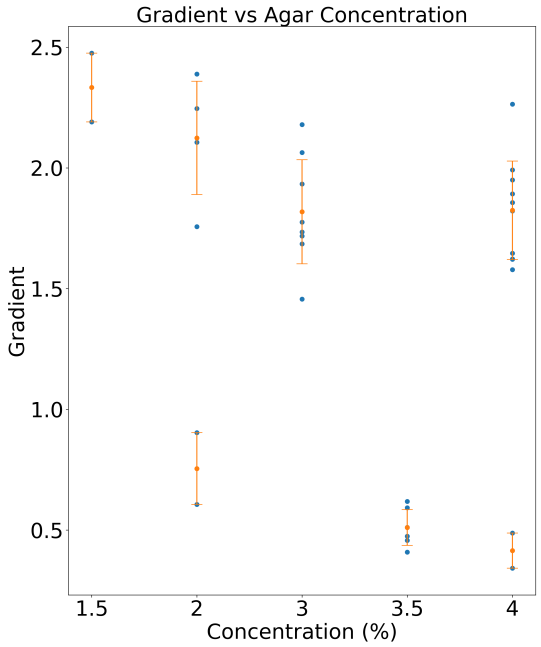



Figure 7: The gradient of the linear part of the first snake versus agarose concentration. For the fast growing colonies the agar concentration are 1.5%, 2%, 3% and 4%, which are the higher four groups of data. The lower three groups of data with concentration 2%, 3.5% and 4% are the slow growing ones. Plotted in orange are the average and error bars.

the growth speed of the colony depends on the agarose concentration as well. The general trend is that the larger the agarose concentration is, the slower the bacteria grows. This result may come from the elasticity of the agarose, the colonies are more difficult to expand in agarose of higher concentration. But although there is an overall trend of decrease, the slight rise of the growth between 3.5% and 4% agarose in the fast growing groups may indicate that this dependence is more complicated than linear under certain condition, it could also be that the dependence on agarose concentration is only limited up till a certain value, and factor other than the elasticity of agarose may be involved.

As the error from the experimental replicates are much larger than that from the list squares fit, the standard deviation of the are shown as the error bars and the error of fitting for each data point are not shown.

# 6 Discussion

This project investigated how the bacteria growth agrees with the assumed mathematical model and the influence of growth condition.

It was found that the assumed model and layer structure is good in describing the growth of bacteria *E.coli* on agarose. Because with the current material from the experiment, only two or three layers can be tracked, which make it difficult to investigate how deep this layer structure keeps until and what happens next. In current research, confocal fluorescence microscopy were used to observe the structure of the cross section of the bacteria colonies [3] [4], but further, possibly numerical research could be done to investigate the detailed structure.

Due to restriction of the scope of this project, only the first layer is investigated in

detail. Future research can be done to investigate further with these data. For example how the transition factor $k_1$ depend on time, which may indicate the factor that influence this process and reveal the nature of the transition. After the research of $k_1$, the more complicated transition between the second and third layer, i.e. $k_2$, can be investigated. Also, the gradient of the growth of the different layers (the $\lambda$ in equations 7, 8 and 9) seem to be different. What are the sizes of the previous layers when the next layers also remains to be researched into.

# 7 Conclusion

In this project it was found that the growth of bacteria *E.coli* on agarose forms a layered structure which can be described with a mathematical model of coupled differential equations. The speed of growth is related to the concentration of agarose. Two or three layers can be observed and the overall quantity of bacteria grows exponentially at the early stage of the growth, whether this rule keeps being true later on cannot be confirmed due to the difficult of tracking the structure of the deeper layers with the current method. The current experimental data has the potential to be further investigated, further confirm the current model and reveal more of the structure of bacteria growth.

# References


[1] Rosalind Allen. *How bacteria form 3D structures*. 2017. URL: https://www.wiki.ed.ac.uk/pages/viewpage.action?spaceKey=SP%7B%5C&%7Dtitle=How+bacteria+form+3D+structures.

[2] Rosalind Allen and Nikola Ojkic. *Private Communication*. James Clerk Maxwell Building (JCMB) Room 2507, King's Buildings Campus, Edinburgh, 2018.

[3] Diarmuid Padraig Lloyd. "Microscopic studies of surface growing bacterial populations". In: (2015), p. 185. URL: http://hdl.handle.net/1842/10509.

[4] Pin Tzu Su et al. "Bacterial Colony from Two-Dimensional Division to Three-Dimensional Development". In: *PLoS ONE* 7.11 (2012), pp. 1–10. ISSN: 19326203. DOI: 10.1371/journal.pone.0048098.


# 8 Appendix

# A Code

This is the code used to analyse the data and generate figures. The data files should have file names of the suitable form as mentioned in the method section and be in the same folder as the code. To run the code for a certain file, change the file name in line 176 in the code for the wanted name, wild card characters can be used to plot multiple file at a time.

There are different parts in the code, if one wants to run it and get a certain figure, uncomment the corresponding part.

```
import math
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import glob

from scipy.optimize import leastsq



"""
Function
Snake_Splitter:
    Input a filename.
        open and read one file
        split the snakes
    return(xarray,yarray)
        xarray is an array consists of 2 or 3 data of snakes
        yarray is an array consists of 2 or 3 lists of 5-d lists
            :
        [data, which snake this is,f_or_s,pc,repeat]
"""

def Snake_Splitter(filename):
    print(filename)
    """analyze the information in the file name"""
    n=0
    index_of_letter=0
    #global re
    for letter in filename:
        if letter=='_':
            n=n+1
            if n==2:
                f_or_s=filename[index_of_letter+1]
                #print(f_or_s)
            if n==3:
                if filename[index_of_letter+2]=='p':
                    pc=filename[index_of_letter+1]
                elif filename[index_of_letter+3]=='p':
                    pc=filename[index_of_letter+1]+filename[
                        index_of_letter+2]
                elif filename[index_of_letter+4]=='p':
                    pc=filename[index_of_letter+1]+filename[
                        index_of_letter+2]+filename[
                        index_of_letter+3]
                #print(pc)
```

```
        if n==4:

            re=filename[index_of_letter+1]
            if index_of_letter+7==len(filename):
                #print('re two digits')
                re=re+filename[index_of_letter+2]
    index_of_letter=index_of_letter+1


#read the file
filein=open(filename,"r")
lines=filein.readlines()
filein.close()

#count the number of snakes
index=0          #the total number of lines in the file
snakenumber=0    #number of snakes
startpoints=[]   #the index of the line that start to be the
   actual data

for line in lines:
    index=index+1
    if line=='0\n':
        startpoints.append(index)
        snakenumber=snakenumber+1

#end points of the snakes
endpoints=[]
for n in range(1,snakenumber):#the first startpoint-2 isn't
   end of any snakes, thus start from startpoint[1].
    endpoints.append(startpoints[n]-2)
endpoints.append(index)


xarray=[]
yarray=[]
#snakes
for index_of_snake in range(0,len(startpoints)):
    n=0
    snakeline=[]
    for n in range(startpoints[index_of_snake],endpoints[
       index_of_snake]):
        snakeline.append(lines[n])

    splitted=[]
    for line in snakeline:
```

```
            splitted.append(line.split("\t"))

        xlist=[]
        ylist=[]
        for line in splitted:
            y=float(line[2])*0.06449
            y=(y**2)*math.pi
            ylist.append([y,index_of_snake+1,f_or_s,pc,re])
            #y is data in area: um^2
            #ylist is a list consist of lists: [data,index of
                snake,...]
            x=float(line[3])*3/60
            xlist.append(x)
        xarray.append(xlist)
        yarray.append(ylist)

    return(xarray,yarray)
    #xarray is an array consists of 2 or 3 data of snakes
    #yarray is an array consists of 2 or 3 lists of 5-d lists


"""
Function
Fitting:
    Input xlist and ylist
        does least square fit,
    plot it
    return k, b values.
"""

def Fitting(X,Y):
    def func(p,x):
        k,b=p
        return k*x+b

    def error(p,x,y):
        return func(p,x)-y

    p0=[1,20]

    Para=leastsq(error,p0,args=(X,Y))

    k,b=Para[0]
    #print("k=",k,"b=",b)
    #print("cost "+str(Para[1]))
    #print("The fitted line:")
```

```
    #print("y="+str(round(k,2))+"x+"+str(round(b,2)))
    """
    plt.figure(figsize=(8,6))#ratio of graph
    plt.scatter(X,Y,color="green",label="data",linewidth=2)

    n=0
    while max(X)>n:
        #this n will be used to determine
        #the range of the x axis of the plot later on.
        n=n+1

    #Plotting the fitted line.
    x=np.linspace(0,n,100) ##Draw 100 dots between the first two
        numbers.
    y=k*x+b ##The function

    plt.plot(x,y,color="red",label="fitted line",linewidth=2)
    plt.legend(loc='lower right')
    plt.xlabel('Time(h)')
    plt.ylabel('Area ($\mu$$m^2$)')
    plt.title('Least Square Fit')
    plt.show()
    """
    return(k,b)




"""main{}"""

"""
Make the data arrays
"""

list_of_blue=['#4dffff','#33bbff','#0040ff','#28288a']
list_of_orange=['#ffea00','#ffbf00','#e67300','#ff4000']
list_of_green=['#99e600','#3bb300','#006600','#003d10']
list_of_red=['#ffb3b3','#ff6666','#ff1f1f','#d60000','#8f0000']
# if only one red is needed, use #ff0000 (h=0,s=100,sl=50)
xarray=[]
yarray=[]

#filename=[]
```

```
filename=glob.glob(r'*_snakes_f_3pc_1.txt')
#variable "filename" is the list of file names read in by "glob"
#wildcard character * and ? are used, e.g.: filename=glob.glob(r
    '*_snakes_?_4pc_?.txt')
for file_index in range(0,len(filename)):
    x,y=Snake_Splitter(filename[file_index])
    #loop around all files of which the file name are
    #in the list "filename".

    #then collect all x and ys of these files.

    for n in range(0,len(x)):
        #can't just xarray.append(x)
        #as then the whole x is one element in xarray.
        xarray.append(x[n])
        yarray.append(y[n])
        #e.g. *_snakes_f_3pc_*.txt
        #xarray=[[1,2,3,...]
                #[1,2,3,...]
                #[1,2,3,...]]
        #yarray=[[[1,1,'f','3','4'],[2,1,f,3,4],[3,1,f,3,4],...]
                #[[1,2,f,3,4],[2,2,f,3,4],[3,2,f,3,4],...]
                #[[1,3,f,3,4],[.........],[.........],...]]
#e.g. print(yarray[0][0][0]) give:2.090898835624298


"""
log the data and append them in yarray (as when doing linear
fitting we need the logged data, the original data are of
course not linear):
"""

for n in range(0,len(yarray)):
    for m in range(0,len(yarray[n])):
        yarray[n][m].append(math.log(yarray[n][m][0]))
#now yarray[0][0][]:
#   0       1           2       3     4          5
#[data,which snake,'fast/slow','pc','no.',data after log]



"""
Linear fitting of the linear part of the first snake
"""

"""pick out the first snakes and the end points"""
```

```
X1=[]
Y1=[]
end_x1=[]
number_of_snake1=0
for n in range(0,len(xarray)):
    #print('snake1:  ',yarray[n][0])
    index_of_snake=yarray[n][0][1]
    if index_of_snake==1:
        number_of_snake1=number_of_snake1+1
        X1.append(xarray[n])
        Y=[]
        for m in range(0,len(yarray[n])):
            Y.append(yarray[n][m])
        Y1.append(yarray[n])
        #print(len(X1[0]),len(Y1[0]))
    #X1 and Y1 are the data matrices for the first snakes
    elif index_of_snake==2:
        end_x1.append(xarray[n][0])


#print(end_x1)
end_points=[]
for n in range(0,len(X1)):
    end_point=0
    for m in range(0,len(X1[n])):
        if X1[n][m]<=end_x1[n]:
            end_point=end_point+1
    end_points.append(end_point)


K=[]
B=[]
xlambda=[]
#print('len(X1):  ',len(X1))
for n in range(0,len(X1)):
    Ytemp=[]
    for m in range(0,len(Y1[n])):
        Ytemp.append(Y1[n][m][5])
    xlambda.append(Y1[n][0][3])
    """cut off from the beginning of the second snake"""
    xcut=[]
    ycut=[]
    for i in range(0,end_points[n]):
        xcut.append(X1[n][i])
        ycut.append(Ytemp[i])
    xcutarray=np.array(xcut)
```

```
        ycutarray=np.array(ycut)

        k,b=Fitting(xcutarray,ycutarray)
        K.append(k)
        B.append(b)
        #print('n:  ',n)
#trusting that the order of the K,B and the order that
    corresponding file shows up in yarray is the same.
which_kb=0
for n in range(0,len(xarray)):
        if yarray[n][0][1]==1:
            #print(which_kb)
            #print('this is the first snake')
            for m in range(0,len(yarray[n])):
                yarray[n][m].append(K[which_kb])
                yarray[n][m].append(B[which_kb])
            which_kb=which_kb+1

#for n in range(0,len(yarray)):
    #if yarray[n][0][1]==1:
        #print('yarray[n][0]:  ',yarray[n][0])

#now yarray[0][0][]:
#   0        1            2        3     4          5         6 7
#[data,which snake,'fast/slow','pc','no.',data after log,k,b]
# only the datas for the first snake
# had been attatched the ks and bs.

#print('K:  ',K)#K is the list of K values
#print('B:  ',B)#B is the list of B values
#print(xlambda)#xlambda is the list of their x values


'''
"""Average and error bar of gradient and intercept"""
x_average=[]# the x list for the average
k_error=[]# ylist for error of k
b_error=[]# ylist for error of b
k_temp=[]# the list that stores the list of k correspond to same
    pc
b_temp=[]
k_average=[]#the list that store the sum of the k with same pc
b_average=[]#the list that store the sum of the b with same pc

for n in range(0,len(xlambda)):
    k=K[n]
```

```
    b=B[n]
    if n+1<len(xlambda):# when this isn't the last item in the
        list
        if xlambda[n+1]==xlambda[n]:# if the next would still be
            the same pc
            k_temp.append(k)
            b_temp.append(b)
        elif xlambda[n+1]!=xlambda[n]:        # if the next is a
            different pc
            k_temp.append(k)
            b_temp.append(b)
            k_average.append(np.mean(k_temp))# append the sum
            b_average.append(np.mean(b_temp))
            k_error.append(np.std(k_temp))
            b_error.append(np.std(b_temp))
            x_average.append(xlambda[n])
            k_temp=[]# reset the temporary lists to prepare for
                the next
            b_temp=[]
    elif n+1>=len(xlambda):# for the last item in the x list
        k_temp.append(k)
        b_temp.append(b)
        k_average.append(np.mean(k_temp))
        b_average.append(np.mean(b_temp))
        k_error.append(np.std(k_temp))
        b_error.append(np.std(b_temp))
        x_average.append(xlambda[n])



"""
Now we have 2 list to be plotted on each graph
"""
"""K"""
plt.figure(figsize=(12,15))
plt.tick_params(labelsize=30)
plt.ylabel('Gradient', size=30)
plt.xlabel('Concentration (%)', size=30)
plt.title('Gradient vs Agar Concentration', size=30)
plt.scatter(xlambda,K, color='#1f77b4')

plt.scatter(x_average, k_average, color='#ff7f0e')
plt.errorbar(x_average, k_average, yerr=k_error, color='#ff7f0e',
    fmt='o', capsize=8)
#(add "elinewidth=5" change the width of the verticle line of
    the error bar thicker...)
```

```
#plt.savefig('gradient.png', dpi=300)
plt.show()

"""B"""
plt.figure(figsize=(12,15))
plt.tick_params(labelsize=30)
plt.ylabel('Intercept ($\mu$m^2$)', size=30)
plt.xlabel('Concentration (%)', size=30)
plt.title('Intercept vs Agar Concentration', size=30)
plt.scatter(xlambda,B, color='#1f77b4')

plt.scatter(x_average, b_average, color='#ff7f0e')
plt.errorbar(x_average, b_average, yerr=b_error, color='#ff7f0e',
    fmt='o', capsize=8)
#plt.savefig('intercept.png', dpi=300)
plt.show()
'''



#Reminder: now yarray[0][0][]:
#   0        1              2        3     4          5          6 7
#[data, which snake, 'fast/slow', 'pc', 'no.', data after log, k, b]



'''
#Code for plotting area versus time:
"""Plot different snakes with different colour in log scale"""

plt.figure(figsize=(15,12))
plt.tick_params(labelsize=30)
plt.ylabel('Area ($\mu$m^2$)', size=30)
plt.xlabel('Time (hours)', size=30)
plt.title('Area of Colony vs Time', size=30)

fast_pc_list=['1.5', '2', '3', '4']
slow_pc_list=['2', '3.5', '4']

for n in range(0,len(xarray)):
    if yarray[n][0][1]==1:
        if yarray[n][0][2]=='f':
            colour=list_of_blue[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
```

```
            colour=list_of_blue[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==2:
        if yarray[n][0][2]=='f':
            colour=list_of_orange[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_orange[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==3:
        if yarray[n][0][2]=='f':
            colour=list_of_green[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_green[slow_pc_list.index(yarray[n
                ][0][3])]

    ylist=[]
    for m in range(0,len(yarray[n])):
        ylist.append(yarray[n][m][0])

    #print(colour)
    plt.semilogy(xarray[n],ylist,color=colour)

#plt.savefig('plot_s_final.png',dpi=300)
plt.show()
'''



'''
"""eliminate intercept for the first snake"""

plt.figure(figsize=(15,12))
plt.tick_params(labelsize=30)
plt.ylabel('log Area ($\mu$m^2$)',size=30)
plt.xlabel('Time (hours)',size=30)
plt.title('Area Minus Intercept vs Time',size=30)

fast_pc_list=['1.5','2','3','4']
slow_pc_list=['2','3.5','4']

for n in range(0,len(xarray)):
    if yarray[n][0][1]==1:
        if yarray[n][0][2]=='f':
            colour=list_of_blue[fast_pc_list.index(yarray[n
```

```
                ][0][3])]
        else:
            colour=list_of_blue[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==2:
        if yarray[n][0][2]=='f':
            colour=list_of_orange[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_orange[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==3:
        if yarray[n][0][2]=='f':
            colour=list_of_green[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_green[slow_pc_list.index(yarray[n
                ][0][3])]

    ylist=[]
    if yarray[n][0][1]==1:
        #print(yarray[n][0])
        for m in range(0,len(yarray[n])):
            ylist.append(yarray[n][m][5]-yarray[n][m][7])
    else:
        for m in range(0,len(yarray[n])):
            ylist.append(yarray[n][m][5])
    plt.plot(xarray[n],ylist,color=colour)

#plt.savefig('plot_all_minus_intercept.png',dpi=300)
plt.show()# the 2nd and 3rd snakes are undisturbed
'''



"""Summing up three snakes for each file"""
#Reminder: now yarray[0][0][]:
#   0       1           2       3     4          5          6 7
#[data,which snake,'fast/slow','pc','no.',data after log,k,b]


xsumlist=[]#the xlist for the sum
ysumlist=[]

for n in range(0,len(xarray)):
```

```
if yarray[n][0][1]==1:#when the current one is the 1st snake
    in the file
    #print('n:  ',n)
    #print('len(yarray):  ',len(yarray))
    if n+2<=len(yarray):#can't be the last list (maybe not
        needed)
        if n+2>=len(yarray) or yarray[n+2][0][1]!=3:# if
            there are 2 snakes
            print('2 snakes')
            #print(yarray[n+2][0][1])
            xend=min(xarray[n][len(xarray[n])-1],xarray[n
                +1][len(xarray[n+1])-1])
            xtemplist=[]
            ytemplist=[]
            for m in range(0,len(xarray[n])):
                if xarray[n][m]<=xend:
                    #print(xarray[n][m])
                    #print(m)
                    xtemplist.append(xarray[n][m])
                    # sum up
                    if xarray[n][m]<xarray[n+1][0]:# before
                        the second snake start
                        yitem=yarray[n][m]
                    elif xarray[n][m]>=xarray[n+1][0]:#
                        after the second snake shows up
                        #print(m)
                        second=0
                        for i in range(0,len(xarray[n+1])):
                            if xarray[n+1][i]<xarray[n][m]:
                                second=second+1# find the
                                    index for which xarray[n
                                    +1][]>=xarray[n][m]
                        #print(second)
                        y=yarray[n][m][0]+yarray[n+1][second
                            ][0]#sum up

                        yitem=[y,'sum',yarray[n][m][2],
                            yarray[n][m][3],yarray[n][m][4],
                            math.log(y),yarray[n][m][6],
                            yarray[n][m][7]]
                    ytemplist.append(yitem)
            #print('i:  ',i)
            #print('xarray[n+1][i]:  ',xarray[n+1][i])
            #print(xarray[n])
            #print(xarray[n+1])
```

```
        #print(n+1)
        #print(yarray[n+1][i][0])
        #print('yarray[n+1][0][0]:  ',yarray[n+1][0][0])
        ysumlist.append(ytemplist)
        xsumlist.append(xtemplist)
    if n+2<len(yarray) and yarray[n+2][0][1]==3:# if
       there are 3 snakes
        #print('3 snakes')
        #print(yarray[n+2][0][1])
        xend=min(xarray[n][len(xarray[n])-1],xarray[n
           +1][len(xarray[n+1])-1],xarray[n+2][len(
           xarray[n+2])-1])
        xtemplist=[]
        ytemplist=[]
        for m in range(0,len(xarray[n])):
            if xarray[n][m]<=xend:
                #print(xarray[n][m])
                #print(m)
                xtemplist.append(xarray[n][m])

                if xarray[n][m]<xarray[n+1][0]:# before
                   the second snake start
                    yitem=yarray[n][m]
                elif xarray[n+2][0]>xarray[n][m]>=xarray
                   [n+1][0]:# after the second, before
                   the third
                    second=0
                    for i in range(0,len(xarray[n+1])):
                        if xarray[n+1][i]<xarray[n][m]:
                            second=second+1# find the
                               index for which xarray[n
                               +1][]>=xarray[n][m]
                    y=yarray[n][m][0]+yarray[n+1][second
                       ][0]
                    yitem=[y,'sum',yarray[n][m][2],
                       yarray[n][m][3],yarray[n][m][4],
                       math.log(y),yarray[n][m][6],
                       yarray[n][m][7]]
                elif xarray[n][m]>=xarray[n+2][0]:
                    second=0
                    third=0
                    for i in range(0,len(xarray[n+1])):
                        if xarray[n+1][i]<xarray[n][m]:
                            second=second+1# find the
                               index for which xarray[n
                               +1][]>=xarray[n][m]
```

```
                        for j in range(0,len(xarray[n+2])):
                            if xarray[n+2][j]<xarray[n][m]:
                                third=third+1# find the
                                    index for which ..., for
                                    the 3rd snake
                        #print(third)
                        y=yarray[n][m][0]+yarray[n+1][second
                            ][0]+yarray[n+2][third][0]
                        yitem=[y,'sum',yarray[n][m][2],
                            yarray[n][m][3],yarray[n][m][4],
                            math.log(y),yarray[n][m][6],
                            yarray[n][m][7]]
                    ytemplist.append(yitem)
                ysumlist.append(ytemplist)
                xsumlist.append(xtemplist)



plt.figure(figsize=(15,12))
plt.tick_params(labelsize=30)
plt.ylabel('Area ($\mu$$m^2$)',size=30)
plt.xlabel('Time (hours)',size=30)
plt.title('Area of Colony vs Time',size=30)

fast_pc_list=['1.5','2','3','4']
slow_pc_list=['2','3.5','4']
pc_list=['1.5','2','3','3.5','4']
for n in range(0,len(xarray)):
    if yarray[n][0][1]==1:
        if yarray[n][0][2]=='f':
            colour=list_of_blue[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_blue[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==2:
        if yarray[n][0][2]=='f':
            colour=list_of_orange[fast_pc_list.index(yarray[n
                ][0][3])]
        else:
            colour=list_of_orange[slow_pc_list.index(yarray[n
                ][0][3])]
    elif yarray[n][0][1]==3:
        if yarray[n][0][2]=='f':
            colour=list_of_green[fast_pc_list.index(yarray[n
                ][0][3])]
```

```
            else:
                colour=list_of_green[slow_pc_list.index(yarray[n
                    ][0][3])]

        ylist=[]
        for m in range(0,len(yarray[n])):
            ylist.append(yarray[n][m][0])

        #print(colour)
        plt.semilogy(xarray[n],ylist,color=colour)

for n in range(0,len(xsumlist)):
        ylist=[]
        for m in range(0,len(ysumlist[n])):
            ylist.append(ysumlist[n][m][0])
        colour=list_of_red[pc_list.index(ysumlist[n][0][3])]
        plt.semilogy(xsumlist[n],ylist,color=colour)
plt.savefig('plot_07_snakes_f_3pc_1_sum.png',dpi=300)
plt.show()
```

```
'''
"""
standby 1:
log axis
"""
plt.figure(figsize=(12,15))
plt.tick_params(labelsize=30)
plt.ylabel('Area (um^2)',size=30)
plt.xlabel('time (h)',size=30)
plt.title('Area versus time',size=30)

for n in range(0,len(xarray)):
    lines=yarray[n]
    ylist=[]

    for data in lines:
        ylist.append(data[0])
```

```
    index_of_snake=data[1]
    if index_of_snake==1:
        plt.semilogy(xarray[n], ylist, color='#1f77b4')
    elif index_of_snake==2:
        plt.semilogy(xarray[n], ylist, color='#ff7f0e')
    elif index_of_snake==3:
        plt.semilogy(xarray[n], ylist, color='#2ca02c')


plt.show()
'''



'''
"""
standby 2:
log the data instead of the axis
"""
plt.figure(figsize=(12,15))
plt.tick_params(labelsize=30)
plt.ylabel('log Area (um^2)', size=30)
plt.xlabel('time (h)', size=30)
plt.title('Area versus time', size=30)

datalist=[]

for n in range(0,len(xarray)):
    lines=yarray[n]
    ylist=[]

    for data in lines:
        a=data[0]
        a=math.log(a)
        ylist.append(a)

    datalist.append(ylist)
    index_of_snake=data[1]
    if index_of_snake==1:
        plt.plot(xarray[n], ylist, color='#1f77b4')
    elif index_of_snake==2:
        plt.plot(xarray[n], ylist, color='#ff7f0e')
    elif index_of_snake==3:
        plt.plot(xarray[n], ylist, color='#2ca02c')
```

```
plt.show()
'''
```