PHYS280: Midterm Project 2

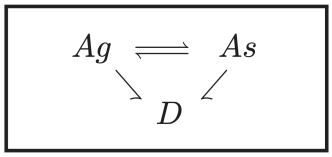
Kavish Senthilkumar

November 11 2021

1 Introduction

In section 10.5.1, we read about a model of kinetochore-microtubule binding. This model uses three states and four reactions.

Specifically, a kinetochore is either attached or detached to a microtubule. When there is an attachment, the complex can exhibit a grow state (represented by the state A_g) or a shrink state (represented by the state A_s). The detached state is called D, and is irreversible. Any attached state can become detached, and attached states can switch, but a detached state remains detached. The possible reactions between states are described below.



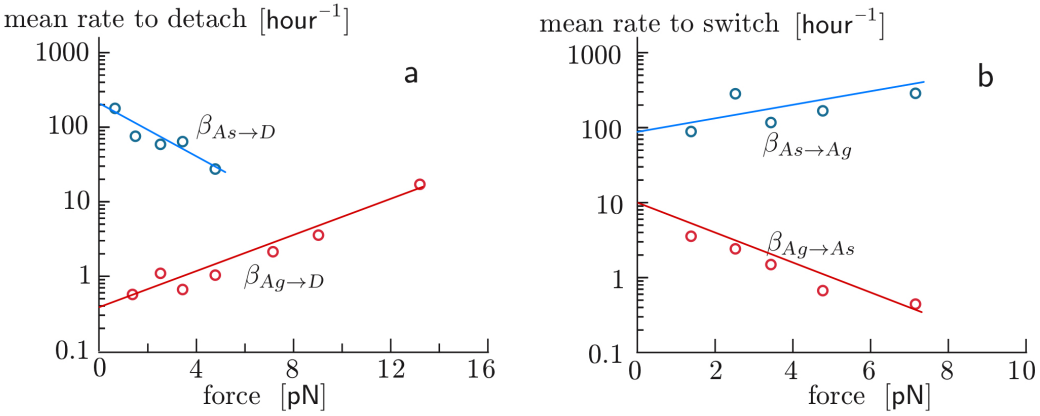
There are rate constants for each of these 4 reactions we must find empirically. Each rate constant depends on the pulling force applied to the kinetochore.

In part A, we calculate each rate constant as a function of applied force. Afterwards, in part B we use these rate constants to run several trials of the Gillespie direct algorithm and estimate PDFs of waiting times for various applied forces from the results. In part C, we graph the mean waiting time for each applied force. In part D, we use the data from part C to consider whether kinetochore-microtubules follow the catch-bond model.

Part A: Calculating rate constants as a function of applied force

To calculate each reaction rate constant β as a function of applied force, we apply exponential fits on the data for every reaction measuring the effect of applied force on mean detachment rate. We use this fit because the data for each reaction appears linear in a semilog plot, indicating an exponential fit is appropriate.

In each calculation F is applied force in pN, and β is a desired rate in hr^{-1}. We graph lines of best fit on the semilog graphs below, and use them to find our exponential fit.



For $\beta_{\mathbf{A_s \to D}}$:

We see at $0N$ of force, $\beta_0 = 150hr^{-1}$. Thus, we look to fit $\beta_{A_s \to D} = 150e^{AF}$. We take another point on the drawn line of best fit on the semilog plot: $(5, 30)$. Substituting, we have $30 = 150e^{5A} \implies A = \frac{-\ln 5}{5}$. Thus, $\boxed{\beta_{A_s \to D} = 150e^{\frac{-\ln 5}{5}F}}$.

For $\beta_{\mathbf{A_g \to D}}$:

We see at $0N$ of force, $\beta_0 = 0.3hr^{-1}$. Thus, we look to fit $\beta_{A_s \to D} = 0.3e^{AF}$. We take another point on the drawn line of best fit on the semilog plot: $(13.75, 15)$. Substituting, we have $15 = .3e^{13.75A} \implies A = \frac{\ln 50}{13.75}$. Thus, $\boxed{\beta_{A_g \to D} = 0.3e^{\frac{\ln 50}{13.75}F}}$.

For $\beta_{\mathbf{A_s \to A_g}}$:

We see at $0N$ of force, $\beta_0 = 90hr^{-1}$. Thus, we look to fit $\beta_{A_s \to D} = 90e^{AF}$. We take another point on the drawn line of best fit on the semilog plot: $(7, 120)$. Substituting, we have $120 = 90e^{7A} \implies A = \frac{\ln(4/3)}{7}$. Thus, $\boxed{\beta_{A_s \to A_g} = 0.3e^{\frac{\ln(4/3)}{7}F}}$.

For $\beta_{\mathbf{A_g \to A_s}}$:

We see at $0N$ of force, $\beta_0 = 10hr^{-1}$. Thus, we look to fit $\beta_{A_s \to D} = 10e^{AF}$. We take another point on the drawn line of best fit on the semilog plot: $(7, 0.3)$. Substituting, we have $.3 = 10e^{7A} \implies A = \frac{\ln(3/100)}{7}$. Thus, $\boxed{\beta_{A_g \to A_s} = 10e^{\frac{\ln(3/100)}{7}F}}$.

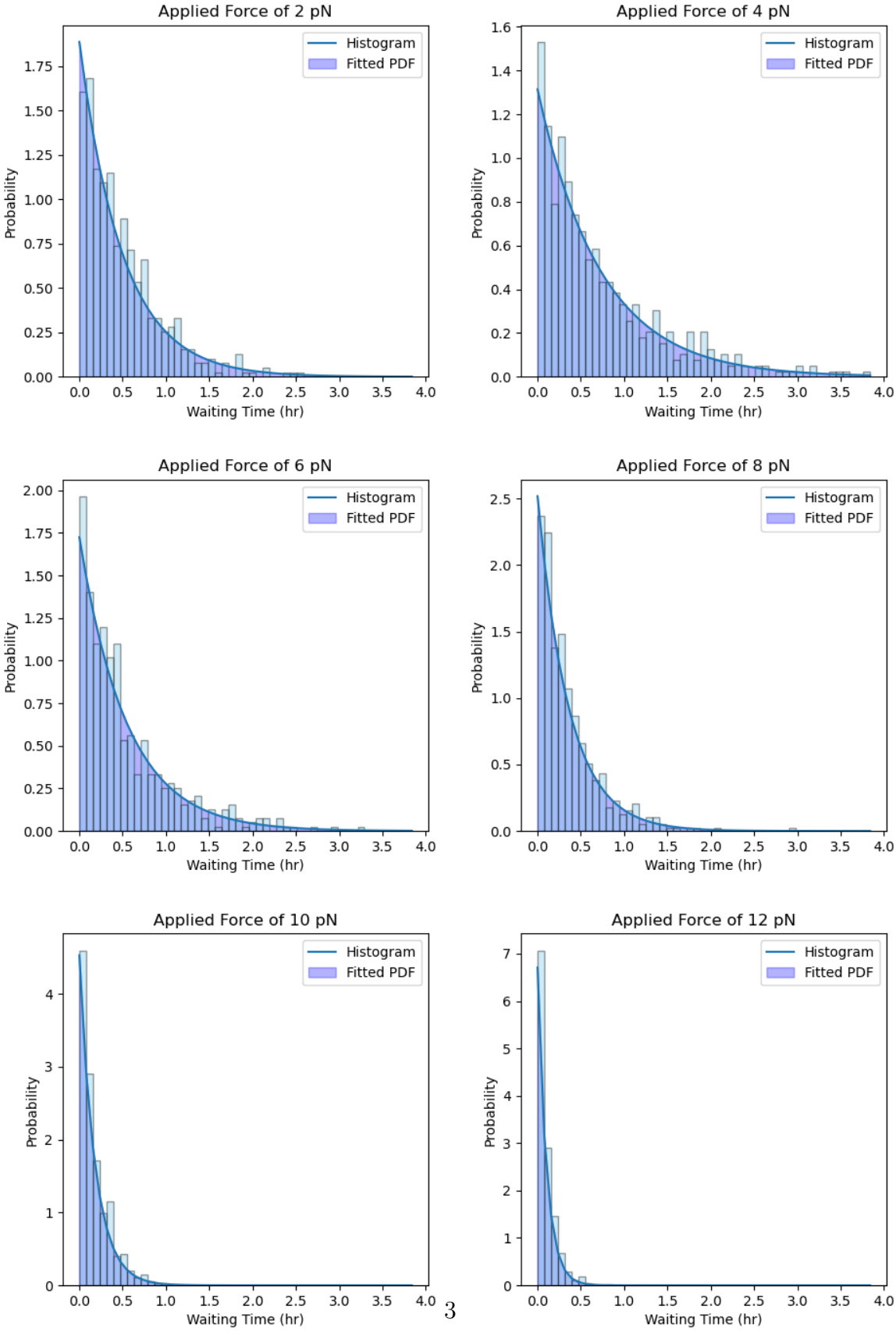
Part B: Using Gillespie direct algorithm to estimate PDFs of wait times for various fixed applied forces

With these rates, we can run multiple iterations of the Gillespie direct algorithm to get a distribution of waiting times until the end state: detachment.

The function `rate_calc` takes in a force in pN as input and uses the equations from part A to calculate a rate constant for every reaction. Then, these rate constants are used in the Gillespie direct algorithm.

We use the `pops` array below to store the state of one kinetochore-microtubule complex, which begins at state A_g at time $t = 0$. We calculate three breakpoints, and account for the irreversibility of the detached state. For each force (we graph PDFs for $2 - 12$ pN in increments of 2 pN) we run the algorithm for 500 trials to create a histogram approximating the PDF. An exponential fit over the histogram is also provided. Code provided in part E.

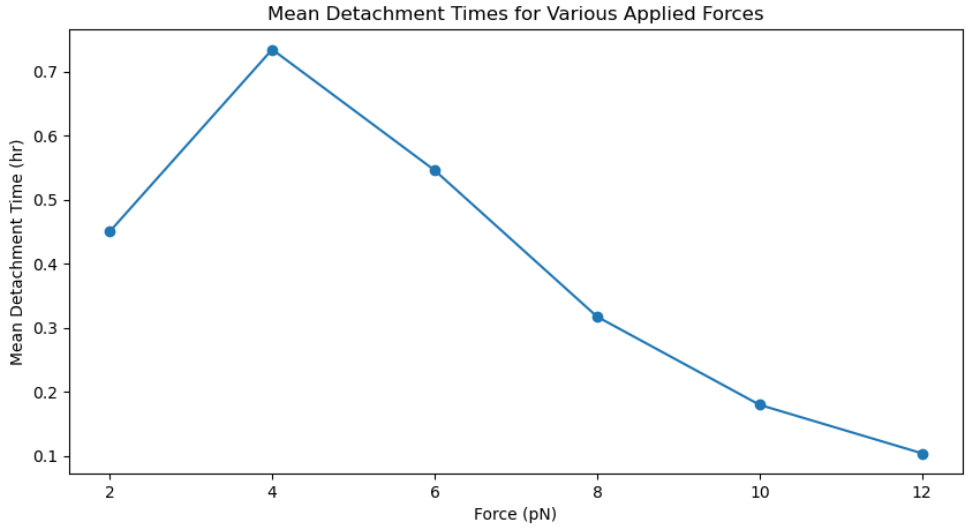
PDFs of Waiting Times for Detachment for Various Applied Forces



Part C: Mean waiting times for each applied force value

The distributions of the PDFs above are exponential, which is consistent with the exponential distribution of the times between events in a Poisson point process like the one we are analyzing. From the shape of the distributions, it appears the rate parameter λ decreases up to 4 pN and then increases.

To verify, we plot the means of waiting times (note the mean is $\frac{1}{\lambda}$) for each value of applied force below. Code provided in part E.



We observe that the mean detachment time increases up to the applied force of 4 pN and then decreases. This is in accordance with the catch-bond model, where an increasing applied force strengthens the bond (increasing mean detachment time) up to a certain point, after which the applied force is large enough that the bond becomes a slip-bond, where an increasing applied force leads to weakening of the bond (decreasing mean detachment time). The point at which the catch-bond begins to act like a slip-bond is apparent in this graph as around 4 pN, when mean detachment time starts decreasing.

Part D: Support of catch-bond model

We conclude that the kinetochore-microtubule bond exhibits catch bonding behavior as per the model outlined in section 10.15.1.

The plot in part C shows that mean detachment time (a measure of bond lifetime) increases up to the applied force of 4 pN, then begins decreasing. Thus, as applied force increases to 4 pN, the bond gets stronger. After 4 pN, mean detachment time (bond lifetime) starts to decrease and the bond resembles a slip-bond. We see a smaller applied force increases bond strength up to a certain value, after which continued increase in force decreases bond lifetime (matching slip-bond behavior). This matches the behavior of a catch-bond.

Part E: Code

Gillespie direct algorithm and PDFs

```
%matplotlib notebook

import numpy as np; import matplotlib.pyplot as plt;
import scipy.stats as scp
from numpy.random import random as rng
from matplotlib import animation
from matplotlib.animation import FuncAnimation
from IPython import display
import math as math

""" Three states:
     0 = Ag; 1 = As; 2 = D
4 reactions:
0: Ag-->D :  rate calculated in rate_calc
1: As-->Ag : rate calculated in rate_calc
2: Ag-->As : rate calculated in rate_calc
3: As-->D :  rate calculated in rate_calc
"""


def rate_calc(F):
    Ag_to_D = 0.3*math.exp((np.log(50)/13.75)*F)
    As_to_Ag = 90*math.exp((np.log(4/3)/7)*F)
    Ag_to_As = 10*math.exp((-1*np.log(100/3)/7)*F)
    As_to_D = 150*math.exp((-1*np.log(5)/5)*F)
    return np.array([Ag_to_D, As_to_Ag, Ag_to_As, As_to_D])

def GDP(F):
    stoich = [[1,0,0], [0,1,0], [1,0,0], [0,1,0]]
    rates = rate_calc(F) # rates in inverse hours
    rates = rates.reshape(4,1)
    pops = np.zeros(3) # track current state in either Ag,As,D
    pops[0] = 1 # starting point in Ag
    ts = list() # make list for waiting times
    while pops[2] == 0:  #while not in detached state, continue
        a = rates
        b = (pops*stoich)
        propens = (a*b).sum(axis=1) #propensities for each rxn
        norm = propens.sum()
        breakpoints = np.cumsum(propens / norm) #calculate three breakpoints
        arr = np.array([0, 0, 0])
        timechooser = np.random.random()  #choose random time and random reaction
        chosen = np.random.random()

        #adjust position based on which breakpoint is chosen
        if (chosen <= breakpoints[0]):
            ts.append(-np.log(timechooser)/norm)
            break
        elif (chosen <= breakpoints[1]):
            arr = np.array([+1, -1, 0])
        elif (chosen <= breakpoints[2]):
            arr = np.array([-1, +1, 0])
        else:
            ts.append(-np.log(timechooser)/norm)
            break
        pops = pops + arr #change to new state
        ts.append(-np.log(timechooser)/norm)# adding waiting time
    return np.sum(ts) #sum up all waiting times
```

```
#Plotting
numtrials = 500
vals = np.zeros((6, numtrials)) #array for each force value we are stimulating
for x in range(numtrials):
    for i in range(6):
        vals[i][x] = GDP(2*(i+1)) #From top to botoom, the data is for 2 pN, 4 pN....12 pN

means = np.apply_along_axis(np.mean, axis=1, arr=vals) #get means


#calculate bins
minimum = [min(vals[0]), min(vals[1]), min(vals[2]), min(vals[3]), min(vals[4]), min(vals[5])]
min_all = min(minimum)
maximum = [max(vals[0]), max(vals[1]), max(vals[2]), max(vals[3]), max(vals[4]), min(vals[5])]
max_all = max(maximum)
bins = np.linspace(min_all, max_all, 50)

#Plot Histograms and PDFs
plt.rcParams['figure.figsize'] = [10, 15]
for i in range(1,7):
    plt.subplot(3,2, i)
    fig = plt.gcf()
    fig.suptitle("PDFs of Waiting Times for Detachment for Various Applied Forces", fontsize=14)
    plt.title("Applied Force of " + str(2*i) +" pN")
    plt.tight_layout(pad=3.0)
    plt.xlabel("Waiting Time (hr)")
    plt.ylabel("Probability")

    ##Fit exponential PDF function, starting at middle of first bin
    num = (max_all-min_all) / 50
    bins2 = np.linspace(min_all+(num/2), max_all, 50)
    loc, scale = scp.expon.fit(vals[i-1])
    y=scp.expon.pdf(bins2, loc, scale)

    #Histogram and PDF below
    plt.plot(bins, y)
    section = np.arange(0, 1, 1/20.)
    plt.fill_between(bins, y, color='blue', alpha=0.3)
    plt.hist(vals[i-1], edgecolor='black', bins=bins, density = True, color="skyblue", alpha = 0.4)
    plt.legend(['Histogram', 'Fitted PDF'])
```

Graph of mean waiting time

```
fig7=plt.figure()
plt.rcParams['figure.figsize'] = [10,5]

#plot means
plt.title("Mean Detachment Times for Various Applied Forces")
plt.xlabel("Force (pN)")
plt.ylabel("Mean Detachment Time (hr)")
plt.plot(["2","4","6","8","10", "12"], means, marker = 'o')
```

Part F: Credits

Sagar for helping change bin size.
Will for using list to keep track of times.
Jas for ideas on finding rate constants as functions of force.
StackExchange for how to fill plot, tips on exponential fit
SciPy stats.expon documentation

Note: This project was honestly pretty cool.