Problem Set 8

December 11, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
```

1 Problem Set 8 - Secret Code 20

1.0.1 8.1A

[55]:
```
#define symbols
k, t = sp.symbols('k t')


n = sp.Matrix([[0, 1]]) #row mtx
w = sp.Matrix([[k*t, 1],[k, 1]]) #weight mtx
c = sp.Matrix([[1],[1]]) #column mtx

N_20_partition_function = n * (w**20) * c
N_20_pf_simplified = sp.simplify(N_20_partition_function[0])
N_20_pf_simplified
```

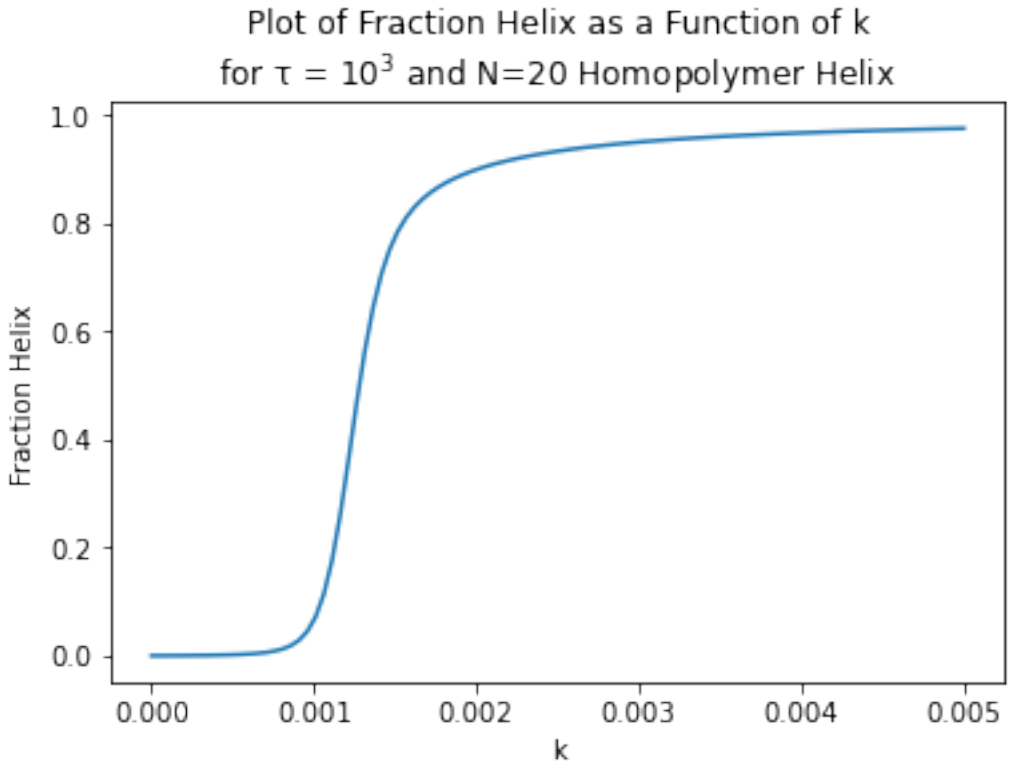
[55]:
$$k^{20}t^{19} + 2k^{19}t^{18} + 18k^{19}t^{17} + 3k^{18}t^{17} + 51k^{18}t^{16} + 136k^{18}t^{15} + 4k^{17}t^{16} + 96k^{17}t^{15} + 480k^{17}t^{14} + 560k^{17}t^{13} + 5k^{16}t^{15} + 150k^{16}t^{14} + 1050k^{16}t^{13} + 2275k^{16}t^{12} + 1365k^{16}t^{11} + 6k^{15}t^{14} + 210k^{15}t^{13} + 1820k^{15}t^{12} + 5460k^{15}t^{11} + 6006k^{15}t^{10} + 2002k^{15}t^{9} + 7k^{14}t^{13} + 273k^{14}t^{12} + 2730k^{14}t^{11} + 10010k^{14}t^{10} + 15015k^{14}t^{9} + 9009k^{14}t^{8} + 1716k^{14}t^{7} + 8k^{13}t^{12} + 336k^{13}t^{11} + 3696k^{13}t^{10} + 15400k^{13}t^{9} + 27720k^{13}t^{8} + 22176k^{13}t^{7} + 7392k^{13}t^{6} + 792k^{13}t^{5} + 9k^{12}t^{11} + 396k^{12}t^{10} + 4620k^{12}t^{9} + 20790k^{12}t^{8} + 41580k^{12}t^{7} + 38808k^{12}t^{6} + 16632k^{12}t^{5} + 2970k^{12}t^{4} + 165k^{12}t^{3} + 10k^{11}t^{10} + 450k^{11}t^{9} + 5400k^{11}t^{8} + 25200k^{11}t^{7} + 52920k^{11}t^{6} + 52920k^{11}t^{5} + 25200k^{11}t^{4} + 5400k^{11}t^{3} + 450k^{11}t^{2} + 10k^{11}t + 11k^{10}t^{9} + 495k^{10}t^{8} + 5940k^{10}t^{7} + 27720k^{10}t^{6} + 58212k^{10}t^{5} + 58212k^{10}t^{4} + 27720k^{10}t^{3} + 5940k^{10}t^{2} + 495k^{10}t + 11k^{10} + 12k^{9}t^{8} + 528k^{9}t^{7} + 6160k^{9}t^{6} + 27720k^{9}t^{5} + 55440k^{9}t^{4} + 51744k^{9}t^{3} + 22176k^{9}t^{2} + 3960k^{9}t + 220k^{9} + 13k^{8}t^{7} + 546k^{8}t^{6} + 6006k^{8}t^{5} + 25025k^{8}t^{4} + 45045k^{8}t^{3} + 36036k^{8}t^{2} + 12012k^{8}t + 1287k^{8} + 14k^{7}t^{6} + 546k^{7}t^{5} + 5460k^{7}t^{4} + 20020k^{7}t^{3} + 30030k^{7}t^{2} + 18018k^{7}t + 3432k^{7} + 15k^{6}t^{5} + 525k^{6}t^{4} + 4550k^{6}t^{3} + 13650k^{6}t^{2} + 15015k^{6}t + 5005k^{6} + 16k^{5}t^{4} + 480k^{5}t^{3} + 3360k^{5}t^{2} + 7280k^{5}t + 4368k^{5} + 17k^{4}t^{3} + 408k^{4}t^{2} + 2040k^{4}t + 2380k^{4} + 18k^{3}t^{2} + 306k^{3}t + 816k^{3} + 19k^{2}t + 171k^{2} + 20k + 1$$

1.0.2 8.1B

[3]:
```
fraction_helix_N_20 = (k/(20*N_20_pf_simplified)) * sp.diff(N_20_pf_simplified,␣
↪k)
lambda_Fh = sp.lambdify([k,t], fraction_helix_N_20)

ks = np.linspace(0, 0.005, 100)
tau = 1e3

plt.plot(ks, lambda_Fh(ks, tau))
plt.title('Plot of Fraction Helix as a Function of k\nfor   = $10^3$ and N=20␣
↪Homopolymer Helix')
plt.xlabel('k')
plt.ylabel('Fraction Helix');
```



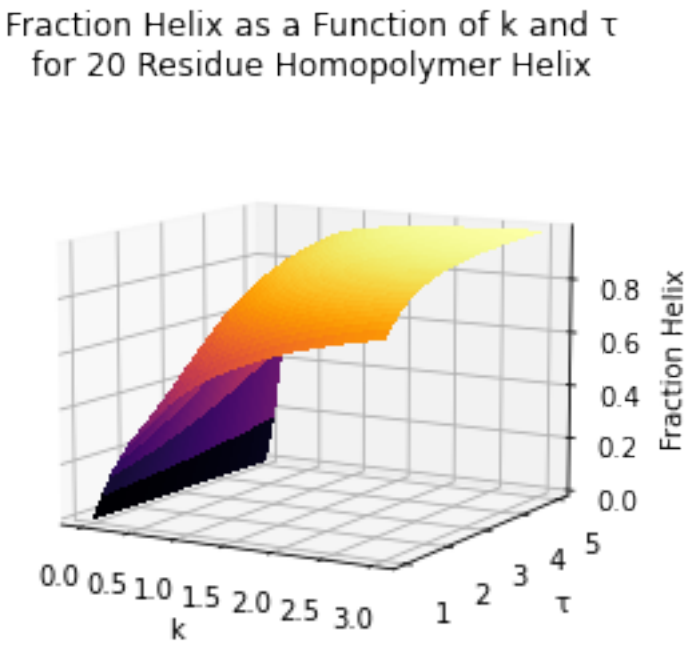
1.0.3 8.1C

This plot does not suffer from the problem we discussed in class for the zipper model because it does not leave out 'islands' or non-continous stretches of helix which were left out in the zipper model. This caused the scooped-out appearance of the zipper model's fraction helix curve when tau = 1 because tau was small enough that the contributions from the islands (which had few or no tau in their boltzmann factors) was no longer insignificant. The matrix model ensures that all possible combinations of helix and non

helix are accounted for in the partition function for the peptide of length N.

[4]:
```
fig = plt.figure()
ax = fig.gca(projection='3d')
ks_3d = np.arange(0, 3.1, .1)
ts_3d = np.arange(1, 5.1, .1)

X, Y = np.meshgrid(ks_3d, ts_3d)
Z = lambda_Fh(X, Y)
surf = ax.plot_surface(X, Y, Z, cmap=cm.inferno, linewidth=0, antialiased=False)
ax.set_xlabel('k')
ax.set_ylabel(' ')
ax.set_zlabel('Fraction Helix')
ax.set_title('Fraction Helix as a Function of k and  \nfor 20 Residue␣
 ↪Homopolymer Helix')
ax.view_init(10, -60)
```

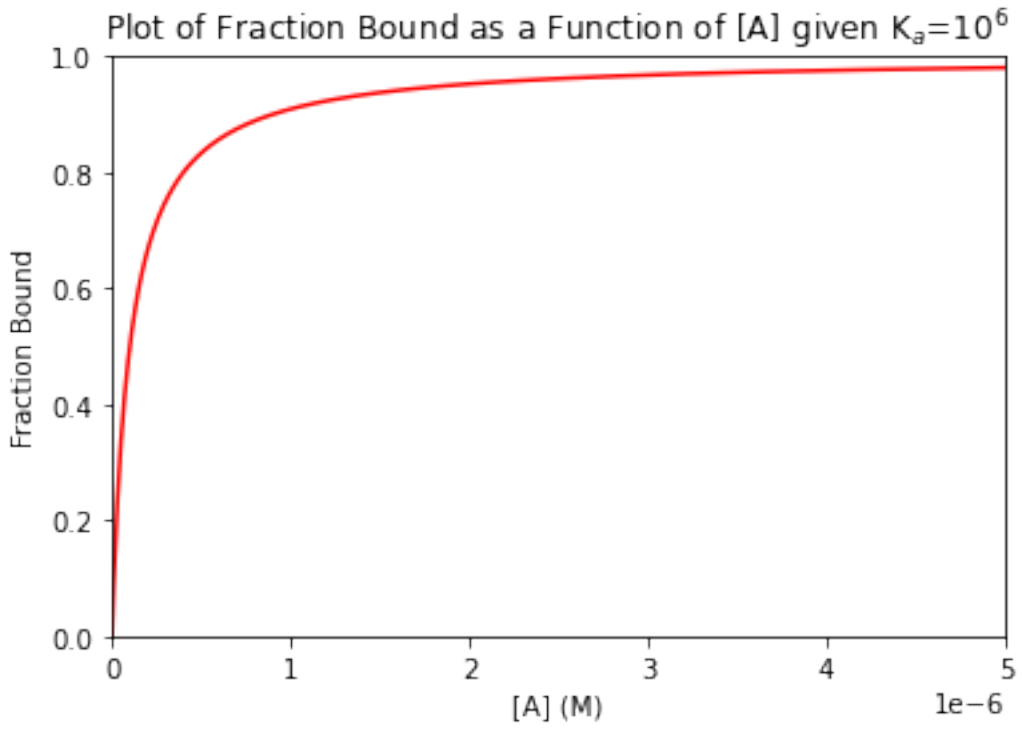


1.0.4 8.2C

[20]:
```
K_a = 10e6 #M^-1
As = np.linspace(0, 1e-3, 100000)
def frac_a(A):
    num = K_a * A
    den = 1+num
    return num/den
```

```
plt.plot(As, frac_a(As), 'r')
plt.xlabel("[A] (M)")
plt.ylabel("Fraction Bound")
plt.title("Plot of Fraction Bound as a Function of [A] given K$_a$=10$^6$ ");
plt.axis([0, 50e-7, 0, 1])
```

[20]: (0.0, 5e-06, 0.0, 1.0)



1.0.5 8.2E

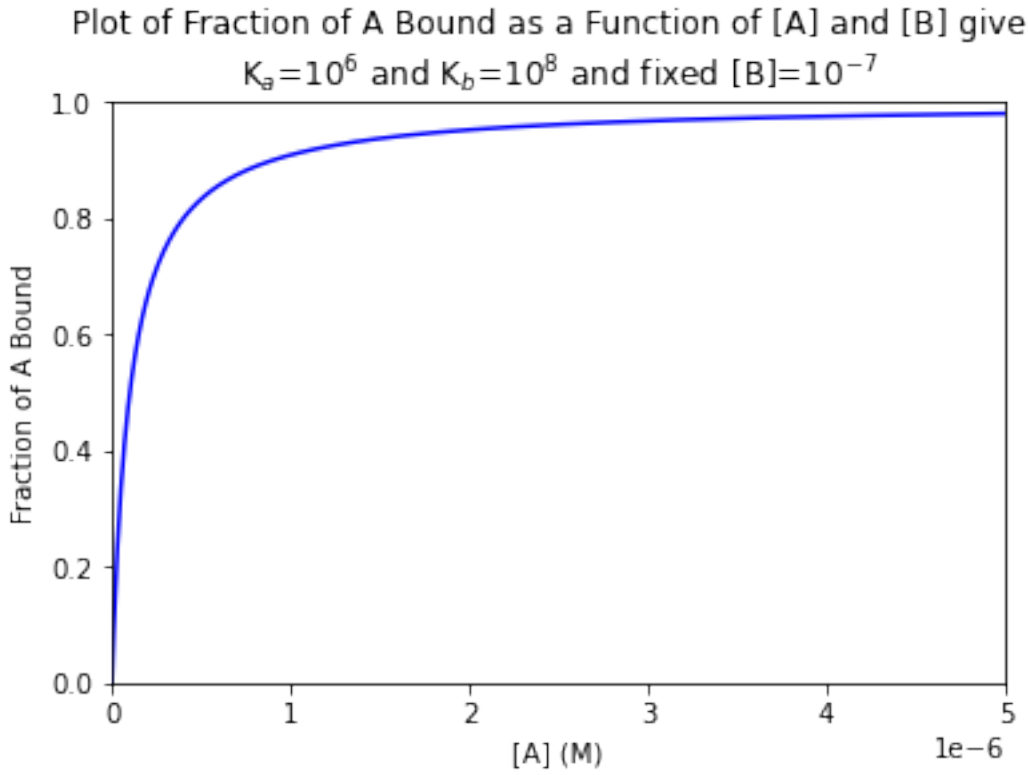
This plot is the same as the plot above which makes sense because the binding of each ligand is completely independent of the other. The cell bellow the plot tests whether the numpy arrays containing the values of the function are essentially equal on a range from 0 (not) to 1 (equal) and it returns 1 indicating that the functions are the same over the input domain

[22]:
```
B = 10e-7
K_b = 10e8

def frac_a_ab(As):
    num = As*K_a + As*B*K_a*K_b
    den = 1 + As*K_a + B*K_b + As*B*K_a*K_b
```

```
    return num/den

plt.plot(As, frac_a_ab(As), 'b')
plt.axis([0, 50e-7, 0, 1])
plt.xlabel("[A] (M)")
plt.ylabel("Fraction of A Bound")
plt.title("Plot of Fraction of A Bound as a Function of [A] and [B] given\nK$_a$=10$^6$ and K$_b$=10$^8$ and fixed [B]=10$^{-7}$");
```



[7]:
```
np.mean(np.isclose(frac_a_ab(As), frac_a(As)))
```

[7]: 1.0

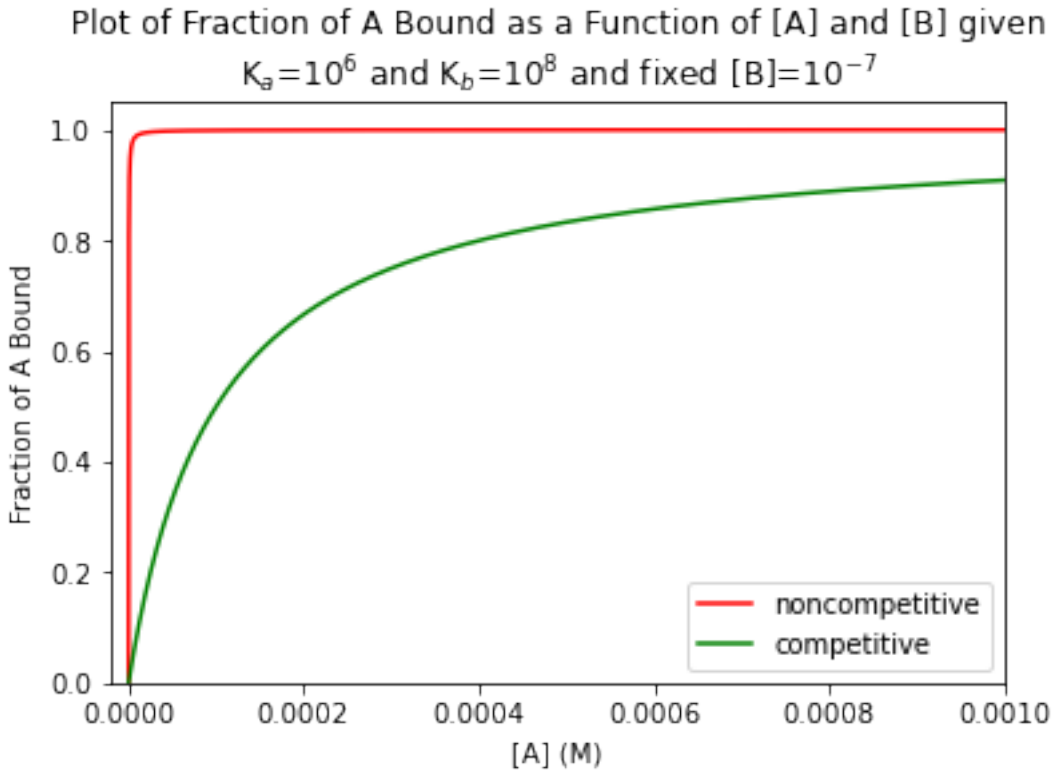
1.0.6 8.2G

The scale between the plot changes so dramatically, I plotted the plot from part C as 'noncompetitive' against the competitive model to show how different the two are. Saturation is significantly more difficult for the competitive macromolecule than the noncompetitive macromolecule. Many orders of magnitude more [A] are required for the competitive macromolecule to even get over 80% saturated with A. The noncompetitive M achieves saturation around 5e-6 M concentration of A while the competitive M doesn't until around 5e-3 M concentration of A, about 1000 times more A.

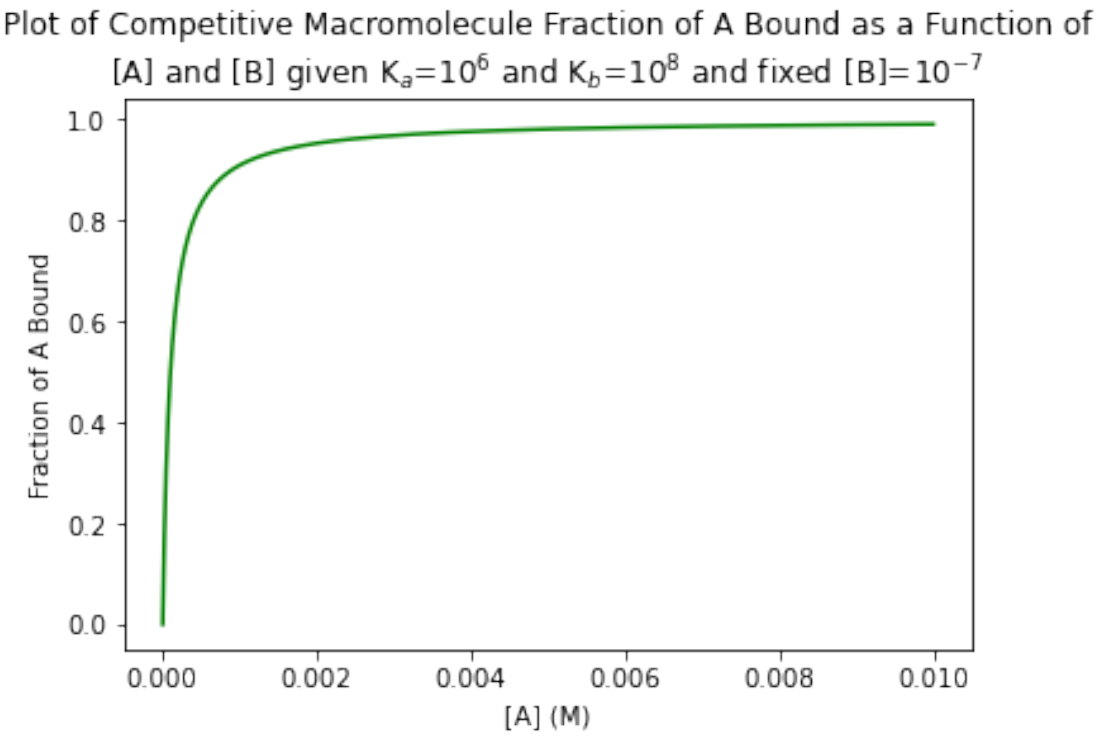
[50]:
```
plt.plot(As, frac_a(As), 'r', label='noncompetitive')

def f_a_comp(As):
    num = K_a*As
    den = 1 + As*K_a + B*K_b
    return num/den

plt.plot(As, f_a_comp(As), 'g', label='competitive')
plt.axis([-2e-5, 1e-3, 0, 1.05])
plt.legend()
plt.xlabel("[A] (M)")
plt.ylabel("Fraction of A Bound")
plt.title("Plot of Fraction of A Bound as a Function of [A] and [B]␣
 ↪given\nK$_a$=10$^6$ and K$_b$=10$^8$ and fixed [B]=10$^{-7}$");
```



[53]:
```
As_extended = np.linspace(0, 1e-2, 100000)
plt.plot(As_extended, f_a_comp(As_extended), 'g')
plt.xlabel("[A] (M)")
plt.ylabel("Fraction of A Bound")
plt.title("Plot of Competitive Macromolecule Fraction of A Bound as a Function␣
 ↪of\n[A] and [B] given K$_a$=10$^6$ and K$_b$=10$^8$ and fixed␣
 ↪[B]=10$^{-7}$");
```


Plot of Competitive Macromolecule Fraction of A Bound as a Function of [A] and [B] given $K_a=10^6$ and $K_b=10^8$ and fixed [B]$=10^{-7}$
Fraction of A Bound
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.002
0.004
0.006
0.008
0.010
[A] (M)