**Problem 1 (Howard 5.2)**

Because the channel is equally likely to be open or closed in the absence of force, it follows that the intrinsic free energy change between the open and closed states is 0, and thus any biasing towards one state or the other is a result of force. Thus:

Plugging in x=2 x 10-9 m and assuming T=298 kelvin, we solve for F and obtain **F = 4.52 pN**

**Problem 2 (Howard 5.4)**

Since the substrate to product ratio is 10-fold of the ratio at equilibrium, we can

write this as:

The energy from converting one molecule of substrate to product can be calculated

as:

Where:

Thus:

For the given ratio of substrate to product, this gives us:

To calculate the total amount of work that could be done with the given substrate to

product ratio, we need to consider that as substrates are converted to product, the

ratio will approach equilibrium. If we define the total number of products and

substrates as:

We can then rewrite the energy gained from converting a single molecule as:

With Keq=1, the total substrate concentration will begin at 10N/11, and approach

N/2. So the total energy gained from the reaction can be written as:

The mean energy gained per molecule is much lower for the whole process. This is

simply because as we approach equilibrium, we can extract less energy for work

from this reaction.

**Problem 3**

(a) The number of particles in state A can only be 0 or 1 since we are considering just 1 particle. Thus, can either be 0 or 1. The probability distribution is discrete, with value PA at and value PB at

We can solve for

(b)

(c) When the number of particles, n, is large, we simply get and , which is what was stated in class.

**Problem 4**

(a) The concentration of oxygen is given by:

In cylindrical coordinates we can take:

Assuming the concentration doesn’t change over z or 𝜃, we can write:

We are given:

Plugging this into the right-side of the equation above, we get:

Thus, this equation for the concentration profile fulfills the steady-state condition of

 (b) We are given:

Additionally, the oxygen consumption rate is given as:

Using the ideal Gas law (), and assuming atmospheric pressure (1atm) and

body temperature(310K), we get:

Thus, the oxygen consumption rate is:

Plotting for the case x=R, with R ranging from 5\*10-6 m to 1.2\*10-4 m, we get:



(c) To find Rcrit, we need to find the point when C = 0. When x= R:

Using MATLAB, we can estimate this to be around . Thus two

capillaries need to be no further than to make sure no tissue

becomes anoxic.

**Problem 5**

We use our Langevin code from last time, included below:

function [out, energies] = Langevin(D, dt, nsteps, T, x0, X, U)

% function out = Langevin(D, dt, nsteps, T, x0, X, U)

% D = diffusion coefficient (m^2/s)

% dt = time step (s)

% nsteps = # of steps to simulate

% T = absolute temperature

% x0 = starting index

% X = vector of positions for the potential

% U = vector containing the potential

% Simulate a trajectory on an arbitrary 1-D potential surface

%AEC 10/15/08

out = zeros(nsteps, 1);

energies = zeros(nsteps, 1);

kb = 1.38 \* 10^(-23);

kT = kb \* T;

gamma = kT/D;

f = -gradient(U)./gradient(X);

out(1) = x0;

dx = X(2) - X(1);

xmin = min(X);

for j = 1:nsteps-1

 fj = f(1 + round((out(j)-xmin)/dx));

 %fj = interp1(X, f, out(j));

 out(j + 1) = out(j) + fj \* dt/gamma + sqrt(2 \* D \* dt) \* randn;

 energies(j+1) = U(1 + round((out(j)-xmin)/dx));

end

Below is the code to implement the double well potential and solve this problem:

**%% Part 1 b (plot trajectory)**

D = 10^(-10); %m^2/s

dt = 10^(-3); %sec

nsteps = 100000; % 100s of data

T = 298; % room temperature

x0 = 0; % start at zero

X = (-1000:1000)\*10^(-8); %a bit wider field of view than suggested to be safe

k= 1.8\*10^(-9); % N/m

a = 6 \* 10^(-16); % N

m = 70; % N/m^3

kb = 1.38 \* 10^(-23); % J / K

kT = kb\*T;

U=a\*X - 0.5\*k\*X.^2 + 0.25\*m\*X.^4; % defining the potential

figure();

x = Langevin(D, dt,nsteps, T, x0, X, U);

ts = dt:dt:100;

plot(ts, 1e6\*x);

xlabel('time (s)');

ylabel('position (um)')

**Sample plot for 10^5 steps:**



**%% Part 1c: Plot U(x) and histogram**

[counts, edges] = histcounts(x);

figure();

plot(X, U/(kb\*T))

bar(edges(1:end-1), counts/(5e2)) % plot scaled histogram

hold on

plot(X, U/(kb\*T))

ylabel('U(x)/k\_BT')

xlabel('position (m)')

**Plot of the energy and probability mass (histogram):**

****

**%% Increase to 5000s of simulation time, so we have enough to estimate a ratio.**

nsteps = 5e6;

x = Langevin(D, dt,nsteps, T, x0, X, U);

[counts, edges] = histcounts(x);

**%% Part d: ratio of population in wells**

du = gradient(U,X);

min\_u = find((du(1:end-1)<0).\*(du(2:end)>0)); %find where dU changes sign. contains indices of where the potential minima are.

wells\_u = U(min\_u); % values of U at the indices of the minima

well\_minima = X(min\_u); % values of X at the indices of the minima (meters)

predicted\_ratio = exp(-(wells\_u(1)-wells\_u(2))/kT);

bins = [find(edges<well\_minima(1),1,'last'),find(edges<well\_minima(2),1,'last')]; %bins corresponding to energy minima

actual\_ratio = counts(bins(1))/counts(bins(2));

fprintf("Predicted ratio p(1)/p(2) from Boltzmann statistics: %3f\n", predicted\_ratio)

fprintf("Actual ratio p(1)/p(2) from simulation: %3f\n", actual\_ratio)

**Predicted ratio p(1)/p(2) from Boltzmann statistics: 4.387929**

**Actual ratio p(1)/p(2) from simulation: 4.423601**

**%% part e: histogram of dwell times**

% Defining the well boundaries.

% We will denote the boundaries of the well as where the second derivative

% changes sign.

Uxx = -k + 3\*m\*X.^2; % second derivative of the potential

well1\_rightborder = find((Uxx(1:end-1)>0).\*(Uxx(2:end)<0));

well2\_leftborder = find((Uxx(1:end-1)<0).\*(Uxx(2:end)>0));

wells\_x = [min(X), X(well1\_rightborder); X(well2\_leftborder), max(X)];

% Wells\_x contains the boundaries of the regions we consider "the wells." We

% plot the regions below.

figure

plot(X, U/kT);

hold on

bar(edges(1:end-1), counts/(1e4)) % plot scaled histogram

xline(wells\_x(1), '--', "well 1")

xline(wells\_x(3), '--')

xline(wells\_x(2), 'r-', "well 2")

xline(wells\_x(4), 'r-')

xlabel("position (m)")

ylabel("U(x)/k\_BT")



% First, we determine which well the particle is in for each step of the trajectory.

function well = which\_well(xx,wells\_x)

% Define the function "which\_well". Returns an array the same length as x

% , which is our trajectory. Each elem has value:

% 0: neither (in between wells)

% 1: in well 1 (left well)

% 2: in well 2 (right well)

w1 = (xx>wells\_x(1,1))\*(xx<wells\_x(1,2));

w2 = (xx>wells\_x(2,1))\*(xx<wells\_x(2,2));

well = w1+2\*w2; % w1 returns 1, w2 returns 2.

End

well\_locations = zeros(size(x));

for i=1:length(x)

 well\_locations(i) = which\_well(x(i),wells\_x);

end

% --- Here's one simple way to get the dwell times, there are probably many

% --- other acceptable solutions!

dwell1 = [];

dwell2 = [];

ctr = 0; % counts the # of timesteps the particle is in a given well for

flag = 0; % denotes the well that we are in. assigned a value the first time we enter that well

for i=1:length(well\_locations) % loop through the array of well locations

 curpos = well\_locations(i); % get current well location

 if curpos ~= flag && curpos ~=0

 % tests if we are in a different well as before.

 if flag == 1

 dwell1 = [dwell1 ctr]; % add the counted time to the dwell times of well 1

 elseif flag == 2

 dwell2 = [dwell2 ctr]; % add the counted time to the dwell times of well 2

 end

 ctr = 0; % reset the counter

 flag = curpos; % set the flag, saying that we are in the new well position

 else

 % otherwise, just increment the counter

 ctr = ctr+1;

 end

end

**%% Make histograms for the fit**

[h1,e1] = histcounts(dwell1\*dt,50);

t1 = 0.5\*(e1(1:end-1)+e1(2:end));

[h2,e2] = histcounts(dwell2\*dt,50);

t2 = 0.5\*(e2(1:end-1)+e2(2:end));

%fit to single exponential

exp\_fit = fittype('a\*exp(-b\*t)','independent','t','coefficients',{'a','b'});

p1 = fit(t1',h1',exp\_fit,'Startpoint',[h1(1),1],'Lower',[0.1\*h1(1),0],'Upper',[10\* h1(1),10^10]);

p2 = fit(t2',h2',exp\_fit,'Startpoint',[h2(1),1],'Lower',[0.1\*h2(1),0],'Upper',[10\* h2(1),10^10]);

%plot histograms

figure

bar(t1,h1)

hold on

plot(t1,p1(t1))

title('Dwell times for well 1')

legend('Data','Fit')

xlabel('Dwell time (s)')

ylabel('Probability mass (proportional to)')

hold off

text(10,50,sprintf('fit: n(t)=a\*exp(-bt) \na=%f \nb=%f (sec^- ^1)',[p1.a,p1.b]));

figure;

bar(t2,h2)

hold on

plot(t2,p2(t2))

title('Dwell times for well 2')

legend('Data','Fit')

xlabel('Dwell time (s)')

ylabel('Probability mass (proportional to)')

hold off

text(4,50,sprintf('fit: n(t)=a\*exp(-bt) \na=%f \nb=%f (sec^- ^1)',[p2.a,p2.b]));

**Dwell times for the “left” well**



**Dwell times for “right” well:**

****

**%% Predicting dwell times for Kramer’s theory**

Uxx = -k + 3\*m\*X.^2; % second derivative to find curvature

alpha\_A = Uxx(min\_u(1));

alpha\_B = Uxx(min\_u(2));

alpha\_TS = abs(Uxx(1001));

k1 = sqrt(alpha\_A\*alpha\_TS)/(2\*pi\*kT/D)\*exp(U(min\_u(1))/kT);

k2 = sqrt(alpha\_B\*alpha\_TS)/(2\*pi\*kT/D)\*exp(U(min\_u(2))/kT);

fprintf("\n Predicted dwell time for well 1: %3f s, k = %3f s^-1", 1/k1, k1)

fprintf("\n Predicted dwell time for well 2: %3f s, k = %3f s^-1 \n", 1/k2, k2)

We can use the formula developed in class to compute the two rates:

To compute the curvatures, we take the second derivative of U at the bottom of each well, and at the transition state. Plugging in the equations, we get:

**Problem 6**

1. The effective diffusion coefficient can be written as:

Where represents the mean square distance travelled along x after

many steps. For N steps, where N is very large:

Since the wavelength of the sinusoid is the distance between the energy

minima.

 can be related to the reaction rates. Imagine a particle in a single energy

well. It will transition to the well in the –x direction with a certain rate , and

it will transition to the adjacent state in the +x direction with the rate. In

the absence of an external force:

Thus, the time elapsed during N steps can be written as:

The factor of 2 in front of comes from the fact that the protein can diffuse

away from a given well both into the left and into the right wells, doubling

the total exit rate.

From Kramers theory, we have:

The curvature of the parabola fits to the wells and peaks of a sinusoid will be

equal and proportional to . D is the free diffusion constant

of the protein. The energy difference between a transition state and a well is

2x the amplitude of the sinusoid (). Thus:

The effective diffusion coefficient is thus:

 (b) Application of a directional force leads to non-equal transition rates to the

minima in the positive and negative x directions. If we define Δ𝑥 as the

distance between adjacent minima, the effect of the force on the reaction rate

is proportional to the distance between an energy minimum and an adjacent

transition point (maximum), which is

The velocity of the particle can be approximated as:

Where () is the net transition rate in the positive direction. Writing

this term out we get:

For a small force, we can expand this term near F = 0 to get a linear

approximation:

To estimate an effective drag coefficient, we make use of the relation:

Here, we can plug in:

And finally get:

 (c) The Stokes-Einstein relation is:

Plugging in our answers for the effective diffusion constant and drag

coefficient from parts (a) & (b):

So we can see that for our effective diffusion constant and drag coefficient, the Stokes-Einstein relation holds.

Intuitively, this only holds for small forces. This is because when the force is large, we can no longer make the approximation that the protein “sits in a well” until it “makes a hop”. Under large force, the well structure effectively disappears and the protein “slides”. Answers that omit the Taylor expansion (and thus didn’t find that the relation holds), but gave an explanation along these lines are accepted.