20.320 Problem Set 2
Question 1

#

In the last problem set, we discovered that Histidine 142 is conserved across many variants of influenza hemagglutinins and thus may play a key role in mediating their pH-dependent conformational change. Now, we will look at the effects the charge of His142 has on the electrostatics of both hemagluttinin conformations.

a) Write a Biopython code to identify the charged residues in both native HA and HA at endosomal pH. As with PS1, consider only residues 40-153 of HA chain B.

b) Write a Biopython code to compute the electrostatic potential of each of the following:

I. Native HA with uncharged His142
II. Native HA with charged His142 (all other histidines uncharged)
III. Endosomal HA with uncharged His142 (all other histidines charged)
IV. Endosomal HA with charged His142

For each case, sum the electrostatic potential of all pairs of charged amino acid using the distance between the charged atoms in your calculations. (Use atom 'OE2' for Glu, 'OD2' for Asp, 'NZ' for Lys, 'NH2' for Arg, and 'NE2' for His) Assume a dielectric constant value of 3.

```
import Bio.PDB
import math
import numpy as np
import pylab

#Histidine is a charged amino acid at endosomal pH
Native_charges = {'GLU':-1.0,'LYS':1.0,'ARG':1.0,'ASP':-1.0}
Endo_charges = {'GLU':-1.0,'LYS':1.0,'ARG':1.0,'ASP':-1.0,'HIS':1.0}
Charged_atom = {'GLU':"OE2",'ASP':"OD2",'LYS':"NZ",'ARG':"NH2",'HIS':"NE2"}

#Parse PDB file for Native HA, find the charged residues and append their indices to
Native_index
native_range = range(39, 153)
Native_charged_residues = []
Native_index = []
for model in Bio.PDB.PDBParser().get_structure("HA_Native", "3EYJ.pdb") :
        Native_polypeptide = Bio.PDB.PPBuilder().build_peptides(model["B"])[0]
        for res_index in native_range:
            if Native_polypeptide[res_index].get_resname() in Native_charges:

Native_charged_residues.append([Native_polypeptide[res_index].get_resname(),
res_index+1])
                Native_index.append(res_index)
print 'Native Charged Residues'
for residue in Native_charged_residues:
        print residue
#Find the position vector of the charged atom in each charged residue
Native_positions = []
for res_index in Native_index:
    atom = Charged_atom[Native_polypeptide[res_index].get_resname()]
```

#

```
Native_positions.append([Native_polypeptide[res_index][atom].coord,Native_polypeptid
e[res_index].get_resname()])

#Parse PDB file for Endosomal HA, find the charged residues and append their indices
to Endo_index
endo_range = range(0, 114)
Endo_charged_residues = []
Endo_index = []
for model in Bio.PDB.PDBParser().get_structure("HA_Endo", "1HTM.pdb") :
        Endo_polypeptide = Bio.PDB.PPBuilder().build_peptides(model["B"])[0]
        for res_index in endo_range:
            if Endo_polypeptide[res_index].get_resname() in Endo_charges:

Endo_charged_residues.append([Endo_polypeptide[res_index].get_resname(),
res_index+40])
                Endo_index.append(res_index)
print 'Endosomal Charged Residues'
for residue in Endo_charged_residues:
        print residue
#Find the position vector of the charged atom in each charged residue
End_positions = []
for res_index in Endo_index:
    atom = Charged_atom[Endo_polypeptide[res_index].get_resname()]

End_positions.append([Endo_polypeptide[res_index][atom].coord,Endo_polypeptide[res
_index].get_resname()])

#Electrostatics for uncharged HIS142 in Native HA
U_total = 0
pairs = 0
for i in range(0,len(Native_positions)):
    for j in range(i+1,len(Native_positions)):
        pairs += 1
        dist_vector = Native_positions[i][0] - Native_positions[j][0]
        distance = np.sqrt(np.sum(dist_vector**2))
        U_total +=
(Native_charges[Native_positions[i][1]]*Native_charges[Native_positions[j][1]])/(3.0*dista
nce)

#Electrostatics for charged HIS142 in Native HA
#Add Histidine 142 to the list of charged residues
Native_positions.append([Native_polypeptide[141]["NE2"].coord,Native_polypeptide[141
].get_resname()])
ch_U_total = 0
ch_pairs = 0
for i in range(0,len(Native_positions)):
    for j in range(i+1,len(Native_positions)):
        ch_pairs += 1
        dist_vector = Native_positions[i][0] - Native_positions[j][0]
        distance = np.sqrt(np.sum(dist_vector**2))
```

```
#
        ch_U_total +=
(Endo_charges[Native_positions[i][1]]*Endo_charges[Native_positions[j][1]])/(3.0*distanc
e)

#Electrostatics for charged HIS142 in Endosomal HA
End_U_total = 0
End_pairs = 0
for i in range(0,len(End_positions)):
    for j in range(i+1,len(End_positions)):
        End_pairs += 1
        dist_vector = End_positions[i][0] - End_positions[j][0]
        distance = np.sqrt(np.sum(dist_vector**2))
        End_U_total +=
(Endo_charges[End_positions[i][1]]*Endo_charges[End_positions[j][1]])/(3.0*distance)

#Electrostatics for uncharged HIS142 in Endosomal HA
#Remove Histidine 142 from the list of charged residues
del End_positions[37]
un_End_U_total = 0
un_End_pairs = 0
for i in range(0,len(End_positions)):
    for j in range(i+1,len(End_positions)):
        un_End_pairs += 1
        dist_vector = End_positions[i][0] - End_positions[j][0]
        distance = np.sqrt(np.sum(dist_vector**2))
        un_End_U_total +=
(Endo_charges[End_positions[i][1]]*Endo_charges[End_positions[j][1]])/(3.0*distance)

print 'Native HA with uncharged His142'
print 'Pairs considered: ' + str(pairs)
print 'U_total = ' + str(U_total)
print 'Native HA with charged His142'
print 'Pairs considered: ' + str(ch_pairs)
print 'U_total = ' + str(ch_U_total)
print 'Endosomal HA with uncharged His142'
print 'Pairs considered: ' + str(un_End_pairs)
print 'U_total = ' + str(un_End_U_total)
print 'Endosomal HA with charged His142'
print 'Pairs considered: ' + str(End_pairs)
print 'U_total = ' + str(End_U_total)
```

#

Output:

Native Charged Residues

['ASP', 46]	['ASP', 79]	['GLU', 120]
['LYS', 51]	['GLU', 81]	['ARG', 121]
['ARG', 54]	['LYS', 82]	['ARG', 123]
['GLU', 57]	['GLU', 85]	['ARG', 124]
['LYS', 58]	['ASP', 86]	['ARG', 127]
['GLU', 61]	['LYS', 88]	['GLU', 128]
['LYS', 62]	['ASP', 90]	['GLU', 131]
['GLU', 67]	['GLU', 97]	['ASP', 132]
['LYS', 68]	['GLU', 103]	['GLU', 139]
['GLU', 69]	['ASP', 109]	['ASP', 145]
['GLU', 71]	['ASP', 112]	['GLU', 150]
['GLU', 74]	['GLU', 114]	['ARG', 153]
['ARG', 76]	['LYS', 117]	

Endosomal Charged Residues

['ASP', 46]	['ASP', 79]	['GLU', 120]
['LYS', 51]	['GLU', 81]	['LYS', 121]
['ARG', 54]	['LYS', 82]	['ARG', 123]
['GLU', 57]	['GLU', 85]	['ARG', 124]
['LYS', 58]	['ASP', 86]	['ARG', 127]
['GLU', 61]	['LYS', 88]	['GLU', 128]
['LYS', 62]	['ASP', 90]	['GLU', 131]
['HIS', 64]	['GLU', 97]	['GLU', 132]
['GLU', 67]	['GLU', 103]	['LYS', 139]
['LYS', 68]	['HIS', 106]	['HIS', 142]
['GLU', 69]	['ASP', 109]	['LYS', 143]
['GLU', 72]	['ASP', 112]	['ASP', 145]
['GLU', 74]	['GLU', 114]	['GLU', 150]
['ARG', 76]	['LYS', 117]	['ARG', 153]

Native HA with uncharged His142
Pairs considered: 703
U_total = 0.0352466471996
Native HA with charged His142
Pairs considered: 741
U_total = -0.090493015543
Endosomal HA with uncharged His142
Pairs considered: 820
U_total = -0.371365158283
Endosomal HA with charged His142
Pairs considered: 861
U_total = -0.588778130675

#

c) Do the calculated electrostatic potentials make sense? If not, why not? How could we improve our energy calculation?

For endosomal HA, the electrostatic potential is lower when His142 is charged. This makes sense because the protonation of histidine at the endosomal pH is important for HA's conformational change that allows the viral and cellular membranes to fuse. Thus endosomal HA is at a lower energy when His142 is protenated.

For native HA, the electrostatic potentials don't make sense because we know histidine is uncharged in the native conformation and thus we would expect that to be the lower energy conformation. One reason that our energy calculations may be off is that we did not include the effect of water molecules.

One area for improvement in our energy calculation would be in choosing which charged atoms we consider. Most of the charged side chains actually exist as resonance structures where the charge is shared among the nitrogens/oxygens. For instance, in glutamate and aspartate, the oxygens in the carboxylic acid groups are partially double-bound to carbon and carry a partial negative charge. Together, the carboxylic acid group carries a charge of -1. For instance, we could assign a charge of -1/2 to each of the oxygens in the side-chains of glutamate and aspartate and consider the position of both in our electrostatic calculations.

20.320 Problem Set 2
Question 2

#

Once again we will be looking at the HA_2 chain (chain 'B' in the PDB files) of Hemagglutinin in its native state and at endosomal pH using the same PDB files from problem set 1.

The conformation potentials used by Chou and Fasman [1] appear in the chart below. These were derived by examining 24 protein structures. The $P_{\alpha/\beta}$ value for each amino acid is proportional to the frequency of that amino acid in alpha helices/beta sheets and has been normalized so that they take on values between zero and two. The amino acids with $P_\alpha > 1$ are assumed to have a propensity for α-helices and similarly those with $P_\beta > 1$ are assumed to have a propensity for β-sheets. Thus Chou and Fasman classified amino acids as strong helix/sheet formers ($H_{\alpha/\beta}$), helix/sheet formers ($h_{\alpha/\beta}$), helix/sheet indifferent (I_α, $i_{\alpha/\beta}$), helix/sheet breakers ($b_{\alpha/\beta}$), and strong helix/sheet breakers ($B_{\alpha/\beta}$). These are also marked in the chart below. In order to understand the Chou-Fasman algorithm, we will use the algorithm to predict alpha-helix propensity in an input protein according to the following rules:

Criteria 1. *Helix Nucleation*. Locate clusters of four helical residues (h_α or H_α) out of six residues along the polypeptide chain. Weak helical residues (I_α) count as $0.5h_\alpha$ (i.e. three h_α and two I_α residues out of six could also nucleate a helix). Helix formation is unfavorable if the segment contains 1/3 or more helix breakers (b_α or B_α).

Criteria 2. *Helix Termination*. Extend the helical segment in *both* directions until terminated by tetrapeptides with P_α,average < 1.00.

Criteria 3. Proline cannot occur in the alpha helix.

For this problem, you will need to download the following files and put them in one folder, including the pdb files for native HA (3EYJ.pdb) and endosomal pH HA (1HTM.pdb):
CFalphaPredict.py
ChouFasman.py

P_α			P_β		
Glu	1.51	H_α	Val	1.70	H_β
Met	1.45		Ile	1.60	
Ala	1.42		Tyr	1.47	
Leu	1.21		Phe	1.38	h_β
Lys	1.16	h_α	Trp	1.37	
Phe	1.13		Leu	1.30	
Gln	1.11		Cys	1.19	
Trp	1.08		Thr	1.19	
Ile	1.08		Gln	1.10	
Val	1.06		Met	1.05	
Asp	1.01	I_α	Arg	0.93	i_β
His	1.00		Asn	0.89	
Arg	0.98	i_α	His	0.87	
Thr	0.83		Ala	0.83	
Ser	0.77		Ser	0.75	b_β
Cys	0.70		Gly	0.75	
Tyr	0.69	b_α	Lys	0.74	
Asn	0.67		Pro	0.55	B_β
Pro	0.57	B_α	Asp	0.54	
Gly	0.57		Glu	0.37	

a) Write the parsePDB function in ChouFasman.py to load chain 'B' of the proteins and return a list of helical residues based on the criteria in PS1 question 1. You must also complete the code for the function responsible for alpha-helix prediction - findAlpha(), in the Python program CFalphaPredict.py according to the rules above. Attach a copy of your code and output. Explain your results.

ChouFasman.py:

```
def parsePDB(fn):
    """
    PDB file parsed to return:outputs:
    seq == list of the protein's amino acids
```

[1] Chou, P; Fasman, G.; "Empirical predictions of protein conformation," *Ann. Rev. Biochem.* **47** (1978) 251-276.

```
#
    actual == list of seq indices to residues known to be in alpha helices
    """
    seq = []
    actual = []

    for model in Bio.PDB.PDBParser().get_structure("File", fn) :
        polypeptides = Bio.PDB.PPBuilder().build_peptides(model["B"])
        for poly_index, poly in enumerate(polypeptides) :
            phi_psi = np.float_(poly.get_phi_psi_list())
            phi_psi_deg = phi_psi * 180 / math.pi
            for res_index, res in enumerate(poly) :
                if res.id[1] >= 40 :
                    if res.id[1] <= 153 :
                        seq.append(res.resname)
                        if phi_psi_deg[res_index, 0] < -57 :
                            if phi_psi_deg[res_index, 0] > -71 :
                                if phi_psi_deg[res_index, 1] > -48 :
                                    if phi_psi_deg[res_index, 1] < -34 :
                                        actual.append(res.id[1])

    print seq
    print actual
    return seq, actual
```

CFalphaPredict.py

```
def P_average(window):
    total = 0.0
    for residue in window:
        total += PA[residue]
    return (total/float(len(window)))


def findAlpha(seq,PA):
    """
    Uses Chau-Fasman criteria to suggest alpha helical regions
    but does not take beta sheets into account
    Inputs:
        seq == (list) the amino acids sequence of the protein
        PA == dictionary whose keys are amino acids and values are the
            CF <Palpha> parameters from the table in your problem set
        PA2 == dictionary of CF a-helix Classifaction for each amino acid
    Outputs:
        AHindices == (list) contains the residue indices of seq that are
            predicted to form helices
    """
    AHindices=[]

    #Search for helix nucleation region
    for i in range(len(seq)-5):
        window = seq[i:i+6]
        if not 'PRO' in window:
```

```
                helix_propensity = 0.0
                breakers = 0
                for aa in window:
                    if PA2[aa] == 'H' or PA2[aa] == 'h':
                        helix_propensity += 1.0
                    if PA2[aa] == 'I':
                        helix_propensity += 0.5
                    if PA2[aa] == 'b' or PA2[aa] == 'B':
                        breakers += 1
                if helix_propensity >= 4.0 and breakers < 2:
                    begin = i
                    end = i+5
                    helix = (begin,end)
                    #Extend nucleation region
                    while (begin-4) >= 0:
                        score = P_average(seq[begin-4:begin])
                        if ('PRO' in seq[begin-4:begin]) or (score < 1.0): break
                        else:
                            begin -= 1 #Extend the nucleation region in the N-term direction
                            helix = (begin,end)
                    while (end+4) < len(seq):
                        score = P_average(seq[end+1:end+5])
                        if ('PRO' in seq[end+1:end+5]) or (score < 1.0): break
                        else:
                            end += 1 #Extend the nucleation region in the C-term direction
                            helix = (begin,end)
                    #Store residues in AH indices
                    for n in range(begin,end+1):
                        if not n in AHindices:
                            AHindices.append(n)
            return AHindices
```

b) Explain the reasons behind the occasional failure of Chou-Fasman alpha-helix predictions.

Chou-Fasman algorithm is based off a very small data set, and therefore is not very representative of all proteins. Also the algorithm is based off of alpha helices and does not include beta sheets. If a beta sheet requirement was included then the algorithm would be more stringent. It also considers only primary structures; therefore if we try to consider the proteins in tertiary or quaternary structure, Chou-Fasman would not be very useful.

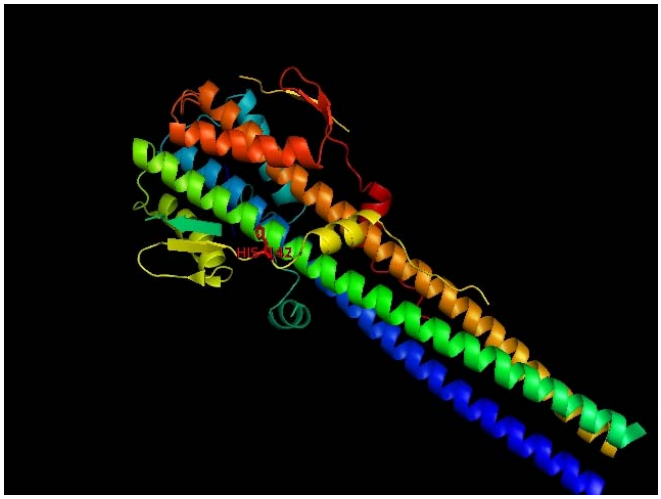
c) Using the PyMOL PDB viewer (zip file attached), look at the structures of these two proteins. Attach print outs of the structures from the viewer. Focusing on the HIS 142 residue, explain the differences in the structures.

20.320 Problem Set 2
Question 2

#

3EYJ:

1HTM:



Note the major difference between the two images with the HIS 142 residue is that in native HA (3EYJ) the HIS is tucked in between the alpha helices, whereas in the endosomal HA (1HTM) the HIS 142 residue is exposed on the side. This explains the accessibility of the HIS 142 residue at different pHs.

20.320 Problem Set 2
Question 3

#

In bacteria, the lactose repressor (*lacI*) is involved with regulating the transcription of genes involved in lactose metabolism. When lactose levels are low, *lacI* is bound to the *lac* operator, preventing the expression of β-galactosidase, which cleaves lactose into its galactose and glucose components. The following sequences were investigated for *lacI* binding in a 1987 paper mapping the recognition helix of *lacI* with the lac operator:

ACTTGTGAGC
ATTTGTGAGC
AAATGTGAGC
AATTGTGAGC
AACTGTGAGC
AATTGTGAGT
AATGGTGAGC
AAGTGTGAGC
AGTTGTGAGC

a) Calculate the $\log_2$(odds) matrix for these sequences. Use pseudocounts of 0.0025 for zero frequencies.

In order to calculate the $\log_2$(odds) matrix, we first create a table with the number of occurrences of each base at each position:

A	9	6	1	0	0	0	0	9	0	0
C	0	1	1	0	0	0	0	0	0	8
G	0	1	1	1	9	0	9	0	9	0
T	0	1	6	8	0	9	0	0	0	1

We can then determine the frequency of each occurrance by dividing the number of occurrences by the total number of sequences – in this case, 9. For each zero frequency, we insert a pseudocount of 0.25%. We must then subtract the sum of the pseudocounts for each nonzero frequency in order to get a total frequency of 1 at each position. This results in the following frequencies matrix:

A	0.9925	0.6667	0.1111	0.0025	0.0025	0.0025	0.0025	0.9925	0.0025	0.0025
C	0.0025	0.1111	0.1111	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.8844
G	0.0025	0.1111	0.1111	0.1106	0.9925	0.0025	0.9925	0.0025	0.9925	0.0025
T	0.0025	0.1111	0.6667	0.8844	0.0025	0.9925	0.0025	0.0025	0.0025	0.1106

In order to calculate the odds of an occurrence at each position, we assume that each base is equally likely to occur at each position. Since there are four bases, we multiply each by 4. This results in the following odds matrix:

A	3.9700	2.6667	0.4444	0.0100	0.0100	0.0100	0.0100	3.9700	0.0100	0.0100
C	0.0100	0.4444	0.4444	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	3.5378
G	0.0100	0.4444	0.4444	0.4422	3.9700	0.0100	3.9700	0.0100	3.9700	0.0100
T	0.0100	0.4444	2.6667	3.5378	0.0100	3.9700	0.0100	0.0100	0.0100	0.4422

#

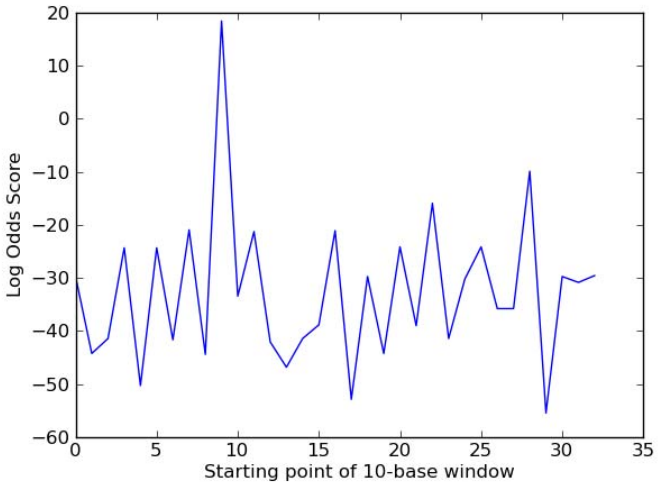
Taking the $\log_2$ of each odds in the above matrix yields:

```
A    1.9891   1.4150  -1.1699  -6.6439  -6.6439  -6.6439  -6.6439   1.9891  -6.6439  -6.6439
C   -6.6439  -1.1699  -1.1699  -6.6439  -6.6439  -6.6439  -6.6439  -6.6439  -6.6439   1.8228
G   -6.6439  -1.1699  -1.1699  -1.1772   1.9891  -6.6439   1.9891  -6.6439   1.9891  -6.6439
T   -6.6439  -1.1699   1.4150   1.8228  -6.6439   1.9891  -6.6439  -6.6439  -6.6439  -1.1772
```

You are interested in a sequence of DNA from a newly discovered organism with an apparently functional lac regulation system. Based on your sequencing results, you predict that *lacI* binds somewhere in the following sequence.

ATCTCATATAATTGTGAGCTCTAATAGAGTTCATGAGCAATG

b) Calculate the $\log_2$(odds) score for each hypothetical binding site in your sequence of interest. Use these values to plot the $\log_2$(odds) score for each 10-base window as a function of the window starting point. For instance, the first value in your plot should be the $\log_2$(odds) score of the sequence ATCTCATATA.



c) Based on your results in Part B, determine the most likely *lacI* binding site in the given sequence. Report the $\log_2$(odds) score of your choice.

The highest $\log_2$(odds) score is 18.41, occurring in a window that begins 9 bases after the beginning of the sequence. Therefore, the *lacI* binding site is likely to be AATTGTGAGC.

20.320 Analysis of Biomolecular and Cellular Systems
Fall 2012