BT2042_Assignment-1

March 12, 2022

1 =============== BT-2042 ===============

1.1 =============== ASSIGNMENT-1 ===============

1.1.1 ENERGY ANALYSIS AT ph = 7

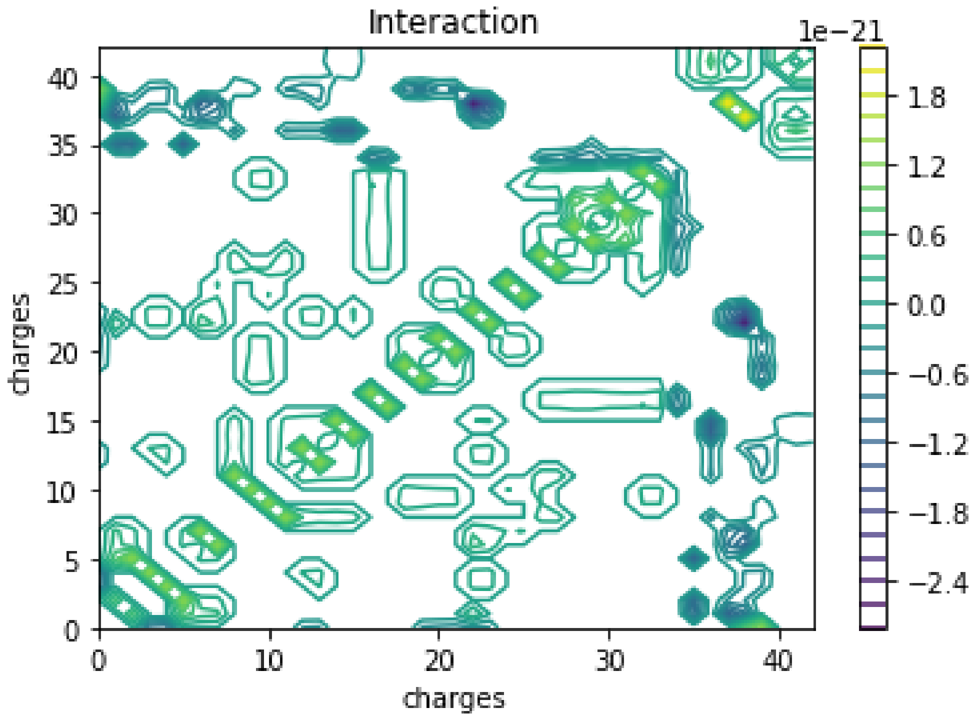
[1]:
```
# required imports

import math as ma
import matplotlib.pyplot as plt
import numpy as np

# residues represents the py file in which we have extracted the charged␣
 ↪residues from the pdb file

from residues import residues_at_ph7
from residues import residues_at_ph2
from residues import residues_at_ph5
from residues import residues_at_ph14
```

[2]:
```
# function to calculate total_energy and distance
# k_new (in a dilectric) = k/(dielectic_contant)
# dist_in_ang respresents distance in argstrong, it needs to be converted to S.
 ↪I units
# assumption particles at distance greater than 15 angstroms are considered␣
 ↪non-interactive
# for sake of ploting we assumed interaction energy between same particles to␣
 ↪be zero.

def cal_energy_and_dist(residues, pot_energy):
    k = 8.98755*10**9
    dielectric_const = 78
    k_new = k/dielectric_const
    total_energy = 0
    for i in range(len(residues) - 1):
        for j in range(i + 1, len(residues)):
            dist_in_ang = ma.sqrt((residues[i]["x"] - residues[j]["x"])**2 + (
                residues[i]["y"] - residues[j]["y"])**2 + (residues[i]["z"] -␣
↪residues[j]["z"])**2)
            if dist_in_ang < 15:
                dist = dist_in_ang*10**-10
                pot_energy[i][j] = (pot_energy[j][i]) = k_new * \
                    residues[i]['charge'] * \
                    residues[j]['charge']/dist
                total_energy += pot_energy[i][j]
    return total_energy
```

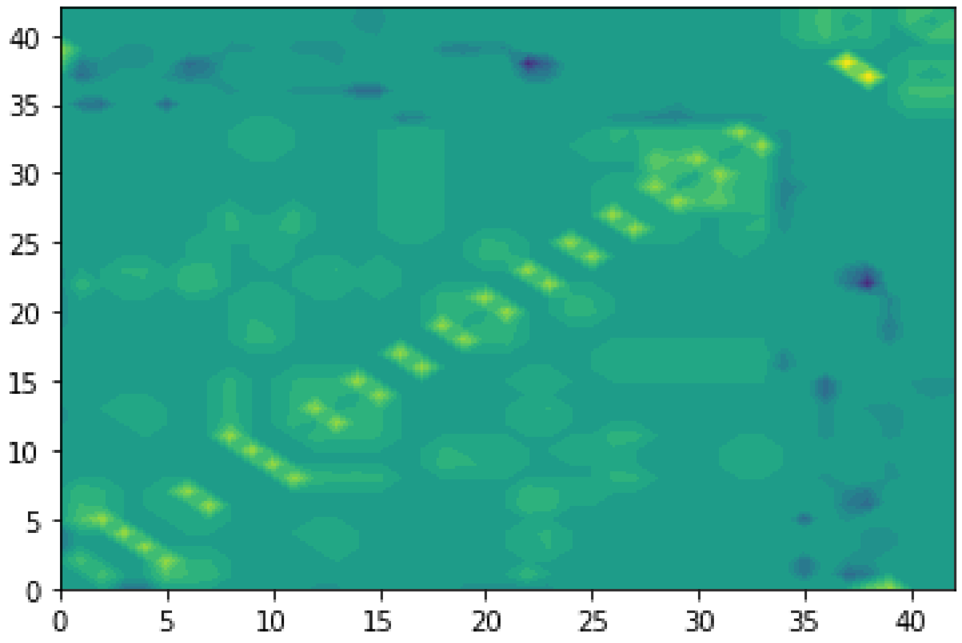
[3]:
```
# analysis at ph 7

print("Total no. of charged-residues_at_ph7 :", len(residues_at_ph7))

# pot_energy_at_ph7 represents matrix of pair wise interaction energy,␣
↪neglecting the pot_enrgy between the same residues_at_ph7
pot_energy_at_ph7 = np.zeros((len(residues_at_ph7), len(residues_at_ph7)))
total_energy_at_ph7 = cal_energy_and_dist(
    residues_at_ph7, pot_energy_at_ph7)

print("TOTAL POTENTIAL ENERGY(at ph 7) : ", total_energy_at_ph7, "Joules")
print("Magnitude of Pairwise INTERACTION ENERGIES(at ph 7) :\n",
      np.abs(pot_energy_at_ph7))
```

```
Total no. of charged-residues_at_ph7 : 43
TOTAL POTENTIAL ENERGY(at ph 7) :  1.7510415187169326e-20 Joules
Magnitude of Pairwise INTERACTION ENERGIES(at ph 7) :
 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 6.42468615e-22 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 6.42468615e-22 0.00000000e+00 … 1.54237037e-22
  1.33791620e-22 1.38538296e-22]
 …
 [0.00000000e+00 0.00000000e+00 1.54237037e-22 … 0.00000000e+00
  5.66206828e-22 5.65888544e-22]
 [0.00000000e+00 0.00000000e+00 1.33791620e-22 … 5.66206828e-22
  0.00000000e+00 5.66363595e-22]
 [0.00000000e+00 0.00000000e+00 1.38538296e-22 … 5.65888544e-22
  5.66363595e-22 0.00000000e+00]]
```

[4]:
```
# pcolor-visualization

plt.contour(pot_energy_at_ph7, 30)
plt.title("Interaction")
plt.xlabel("charges")
plt.ylabel("charges")
plt.colorbar()
plt.show()
```



[5]:
```
plt.contourf(pot_energy_at_ph7, 20 )
plt.show()
```



1.1.2 ENERGY ANALYSIS @ ph = 2, 5, 14

[6]:
```
# analysis at ph 2

print("Total no. of charged-residues_at_ph2 :", len(residues_at_ph2))

# pot_energy_at_ph2 represents matrix of pair wise interaction energy,␣
↪neglecting the pot_enrgy between the same residues_at_ph2
pot_energy_at_ph2 = np.zeros((len(residues_at_ph2), len(residues_at_ph2)))

total_energy_at_ph2 = cal_energy_and_dist(
    residues_at_ph2, pot_energy_at_ph2)

print("TOTAL POTENTIAL ENERGY(at ph 2) : ", total_energy_at_ph2, "Joules")
print("Magnitude of Pairwise INTERACTION ENERGIES(at ph 2) :\n",
      np.abs(pot_energy_at_ph2))
```

```
Total no. of charged-residues_at_ph2 : 16
TOTAL POTENTIAL ENERGY(at ph 2) :  2.778865150235577e-20 Joules
Magnitude of Pairwise INTERACTION ENERGIES(at ph 2) :
 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 7.76819820e-22 1.30128971e-21 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  4.89181665e-22 5.16020727e-22 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 4.45450793e-22 4.81988931e-22]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.33486005e-22
 2.84053629e-22 2.85973351e-22 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.21707722e-22]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.39114795e-22
 6.71968787e-22 5.61902220e-22 4.51893211e-22 4.55864754e-22
 0.00000000e+00 0.00000000e+00 5.53412829e-22 5.55964382e-22]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 2.18068368e-21 0.00000000e+00 2.57533283e-22
 0.00000000e+00 2.70682408e-22 4.72505903e-22 4.02683970e-22
 3.92868994e-22 4.05693539e-22 0.00000000e+00 0.00000000e+00]
[7.76819820e-22 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.18068368e-21 0.00000000e+00 0.00000000e+00 2.72051478e-22
 2.61998260e-22 2.91232600e-22 5.77587034e-22 4.92240315e-22
 5.76167266e-22 6.09332128e-22 0.00000000e+00 0.00000000e+00]
[1.30128971e-21 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 5.23097353e-22 5.16881431e-22 0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 3.33486005e-22 5.39114795e-22
 2.57533283e-22 2.72051478e-22 0.00000000e+00 0.00000000e+00
 5.66206828e-22 5.65888544e-22 1.79682280e-22 1.64495532e-22
 1.52395757e-22 1.55269557e-22 1.92462687e-22 2.00000472e-22]
[0.00000000e+00 0.00000000e+00 2.84053629e-22 6.71968787e-22
 0.00000000e+00 2.61998260e-22 0.00000000e+00 5.66206828e-22
 0.00000000e+00 5.66363595e-22 1.92149542e-22 1.81486317e-22
 1.60345230e-22 1.66076739e-22 2.10492023e-22 2.09466038e-22]
[0.00000000e+00 0.00000000e+00 2.85973351e-22 5.61902220e-22
 2.70682408e-22 2.91232600e-22 0.00000000e+00 5.65888544e-22
 5.66363595e-22 0.00000000e+00 2.24461159e-22 2.02019195e-22
 1.58852108e-22 1.67413009e-22 1.69996309e-22 1.70803120e-22]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 4.51893211e-22
 4.72505903e-22 5.77587034e-22 0.00000000e+00 1.79682280e-22
 1.92149542e-22 2.24461159e-22 0.00000000e+00 1.29809926e-21
 2.55134038e-22 3.05911492e-22 0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 4.55864754e-22
 4.02683970e-22 4.92240315e-22 0.00000000e+00 1.64495532e-22
 1.81486317e-22 2.02019195e-22 1.29809926e-21 0.00000000e+00
 2.48038047e-22 3.00560590e-22 0.00000000e+00 0.00000000e+00]
[4.89181665e-22 0.00000000e+00 0.00000000e+00 0.00000000e+00
 3.92868994e-22 5.76167266e-22 5.23097353e-22 1.52395757e-22
 1.60345230e-22 1.58852108e-22 2.55134038e-22 2.48038047e-22
 0.00000000e+00 1.29778358e-21 2.00234444e-22 0.00000000e+00]
[5.16020727e-22 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.05693539e-22 6.09332128e-22 5.16881431e-22 1.55269557e-22
 1.66076739e-22 1.67413009e-22 3.05911492e-22 3.00560590e-22
 1.29778358e-21 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[0.00000000e+00 4.45450793e-22 0.00000000e+00 5.53412829e-22
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.92462687e-22
 2.10492023e-22 1.69996309e-22 0.00000000e+00 0.00000000e+00
 2.00234444e-22 0.00000000e+00 0.00000000e+00 1.29723104e-21]
[0.00000000e+00 4.81988931e-22 4.21707722e-22 5.55964382e-22
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.00000472e-22
 2.09466038e-22 1.70803120e-22 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.29723104e-21 0.00000000e+00]]
```

[7]:
```
# analysis at ph = 5

print("Total no. of charged-residues_at_ph5 :", len(residues_at_ph5))

# pot_energy_at_ph5 represents matrix of pair wise interaction energy,␣
 ↪neglecting the pot_enrgy between the same residues_at_ph5
pot_energy_at_ph5 = np.zeros((len(residues_at_ph5), len(residues_at_ph5)))

total_energy_at_ph5 = cal_energy_and_dist(
    residues_at_ph5, pot_energy_at_ph5)

print("TOTAL POTENTIAL ENERGY(at ph 5) : ", total_energy_at_ph5, "Joules")
print("Magnitude of Pairwise INTERACTION ENERGIES(at ph 2) :\n",
      np.abs(pot_energy_at_ph5))
```

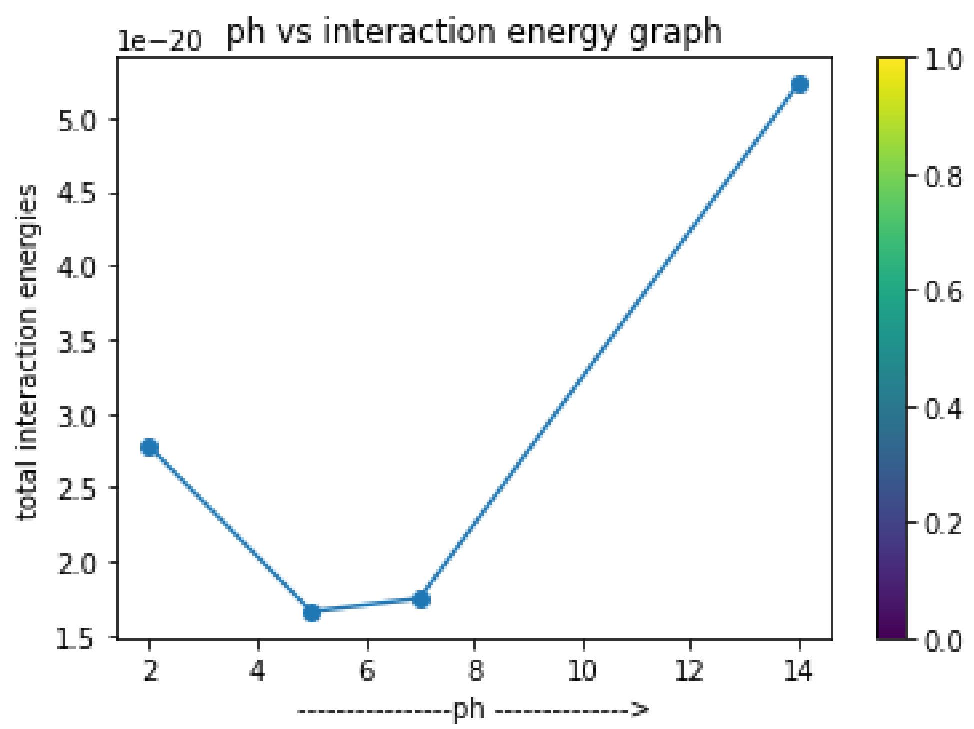
```
Total no. of charged-residues_at_ph5 : 49
TOTAL POTENTIAL ENERGY(at ph 5) :  1.6623913198776668e-20 Joules
Magnitude of Pairwise INTERACTION ENERGIES(at ph 2) :
 [[0.00000000e+00 0.00000000e+00 0.00000000e+00 … 5.16020727e-22
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 6.42468615e-22 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 6.42468615e-22 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 …
 [5.16020727e-22 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 1.29723104e-21]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  1.29723104e-21 0.00000000e+00]]
```

[8]:
```
# analysis at ph = 14

print("Total no. of charged-residues_at_ph14 :", len(residues_at_ph14))
```

```
# pot_energy_at_ph14 represents matrix of pair wise interaction energy,
neglecting the pot_enrgy between the same residues_at_ph14
pot_energy_at_ph14 = np.zeros((len(residues_at_ph14), len(residues_at_ph14)))

total_energy_at_ph14 = cal_energy_and_dist(
    residues_at_ph14, pot_energy_at_ph14)

print("TOTAL POTENTIAL ENERGY(at ph 14) : ", total_energy_at_ph14, "Joules")
print("Magnitude of Pairwise INTERACTION ENERGIES(at ph 14) :\n",
      np.abs(pot_energy_at_ph14))
```

```
Total no. of charged-residues_at_ph14 : 33
TOTAL POTENTIAL ENERGY(at ph 14) :  5.226480088660272e-20 Joules
Magnitude of Pairwise INTERACTION ENERGIES(at ph 14) :
 [[0.00000000e+00 6.42468615e-22 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [6.42468615e-22 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 …
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  3.69382960e-22 3.60257677e-22]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 3.69382960e-22
  0.00000000e+00 1.33670773e-21]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 … 3.60257677e-22
  1.33670773e-21 0.00000000e+00]]
```

[9]:
```
# Ploting and Interaction Energy

energies = [total_energy_at_ph2, total_energy_at_ph5,
            total_energy_at_ph7, total_energy_at_ph14]
ph = [2, 5, 7, 14]
plt.plot(ph, energies)
plt.scatter(ph, energies)
plt.title("ph vs interaction energy graph")
plt.xlabel('----------------ph -------------->')
plt.ylabel('total interaction energies')
plt.colorbar()
plt.show()
```



[10]:
```
# in the graph above it can be observed that the interaction energy obtained at␣
↪ph 5 is minimum.
# hence, at ph 5 our proteien is most stable out of 2, 5, 7, 14
```

1.1.3 Calculation of ISOELECTRIC POINT

[2]:
```
# This is the sequence of the chain A it has been extracted from chain A
sequence = ['MET', 'GLU', 'LEU', 'LYS', 'HIS', 'SER', 'ILE', 'SER', 'ASP',
            'TYR', 'THR', 'GLU', 'ALA', 'GLU', 'PHE', 'LEU', 'GLN', 'LEU',
            'VAL', 'THR', 'THR', 'ILE', 'CYS', 'ASN', 'ALA', 'ASP', 'THR',
            'SER', 'SER', 'GLU', 'GLU', 'GLU', 'LEU', 'VAL', 'LYS', 'LEU',
            'VAL', 'THR', 'HIS', 'PHE', 'GLU','GLU', 'MET', 'THR', 'GLU',
            'HIS', 'PRO', 'SER', 'GLY', 'SER', 'ASP', 'LEU', 'PRO', 'LYS',␣
↪'GLU',
            'GLY', 'ASP', 'ASP', 'ASP', 'SER', 'PRO', 'SER', 'GLY', 'ILE',␣
↪'VAL',
            'ASN', 'THR', 'VAL', 'LYS', 'GLN', 'TRP', 'ARG', 'ALA', 'ALA',␣
↪'ASN',
            'GLY', 'LYS', 'SER', 'GLY', 'PHE', 'LYS', 'GLN', 'GLY']


# here we are calculating frequency of a charged residue in the given chain␣
↪sequence.
```

```python
def count_of_charged_residue(name):
    cnt = 0
    for i in range(len(sequence)):
        if sequence[i] == name:
            cnt += 1
    return cnt


# data represents list of dicts. these dicts contain name frequency and pka␣
 ↪values of the charged residues.
data = [{"name": "ASP", "count": count_of_charged_residue(
    "ASP"), "pka": 3.65}, {"name": "GLU", "count":␣
 ↪count_of_charged_residue("GLU"), "pka": 4.25},
    {"name": "HIS", "count": count_of_charged_residue("HIS"), "pka": 6.00},
    {"name": "LYS", "count": count_of_charged_residue("LYS"), "pka": 10.53},
    {"name": "ARG", "count": count_of_charged_residue("ARG"), "pka": 12.48}]


# these are the critical ph points, they represents ph values where we may␣
 ↪experience charge change.
# 2.34 represents pka1 value of GLY, it is been considered since it ends our␣
 ↪sequence.
# 9.21 represents pka2 value of MET, it is been considered since it starts our␣
 ↪sequence.
critical_ph_points = [2, 3, 4, 5,  8, 11, 13]


# Function to return iso-electronic point of our protien.
# We are iterating over critical_ph_points list and checking at what ph the␣
 ↪sign of the net charge changes.
# Then we calculate the wieghted mean of the pka3 value which surround the␣
 ↪critical ph obtained.
def cal_iso_pt():
    charge = 0
    for ph in critical_ph_points:
        prev_charge = charge
        charge = {ph < 2.34: +1, 2.34 < ph < 9.21: 0}.get(True, -1)
        for item in data:
            if item["name"] in ["ASP", "GLU"]:
                if ph > item["pka"]:
                    charge -= item["count"]
            if item["name"] in ["ARG", "LYS", "HIS"]:
                if ph < item["pka"]:
                    charge += item["count"]
        if charge*prev_charge <= 0:
            if 3.65 < ph < 4.25:
                return ((3.65*data[0]["count"] + 4.25*data[1]["count"]) /␣
 ↪(data[0]["count"] + data[1]["count"]))
            elif 4.25 < ph < 6:
                return ((4.25*data[1]["count"] + 6*data[2]["count"]) /␣
 ↪(data[1]["count"] + data[2]["count"]))
            elif 6 < ph < 10.53:
                return ((6*data[2]["count"] + 10.53*data[3]["count"]) /␣
 ↪(data[2]["count"] + data[3]["count"]))
            elif 10.53 < ph < 12.48:
                return ((10.53*data[3]["count"] + 12.48*data[4]["count"]) /␣
 ↪(data[3]["count"] + data[4]["count"]))


print("Isoelectric point of the given protien is :", cal_iso_pt())
```

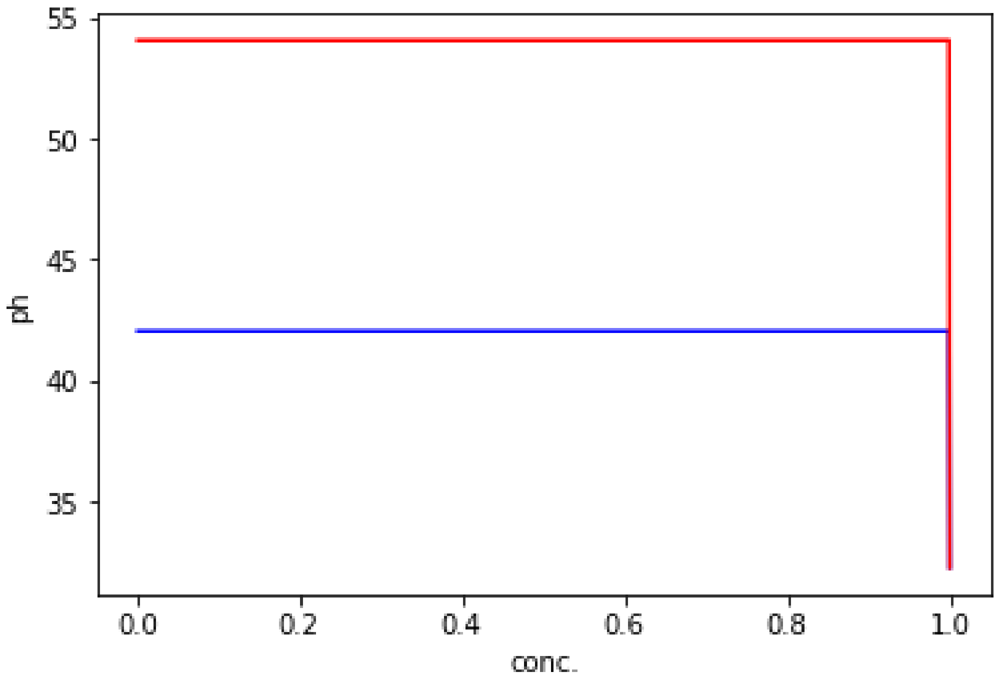
```
Isoelectric point of the given protien is : 4.653846153846154
```

[12]:
```
# the answers obtained is 4.653846153846154, it shows that the iso-electronic␣
 ↪point of our protien.
# the ans may not be accurate as we have calculated the wieghted mean an did'nt␣
 ↪knew the exact algorithm.
```

1.1.4 ph-titration curve

[4]:
```
# required import
from re import A
import numpy as np
import matplotlib.pyplot as plt

kw = 10**-14
y = []
conc = np.linspace(0.001, 1, 999)

# here we are using cubic polynomial for ph


def cal_ph_values(k):
    list = np.zeros(999)
    for i in range(len(conc)):
        coeff = [1, (k + conc[i]/(1 + conc[i])), conc[i]*k /
                 (1 + conc[i]) - k/(1 + conc[i]) + kw, k*kw]
        np.roots(coeff)
        coeff = sorted(x.real for x in coeff)
        list[i] = {coeff[0] > 0: coeff[0], coeff[1]
                   > 0: coeff[1]}.get(True, coeff[2])
    return list
```

```python
# i have considered ka values of glutamic acid
k_values = {"ka1": 10**-4.25, "ka2": 10**-9.47}


plt.plot(conc, -np.log(cal_ph_values(k_values["ka1"])),
         label=f"curve for glu ka1", color="blue")
plt.plot(conc, -np.log(cal_ph_values(k_values["ka2"])),
         label=f"curve for glu ka2", color="red")
plt.xlabel("conc.")
plt.ylabel("ph")
plt.show()
```


ph
conc.