

## Activité 5

# Courbes de répartition

Rien de nouveau d'un point de vue « théorique » cette semaine... normal, vous savez tout !

Mais... il reste l'essentiel qui nous réunit ici : savoir passer de l'exercice 1 à l'exercice 3.

Pour résoudre l'exercice 1, n'importe quel « bricolage » convient. Il suffit, en gros de savoir comment définir une fonction et comment tracer une courbe pour y arriver. Mais, si on reprend la même question dans l'exercice 3, il faudra réfléchir à la façon de modéliser le système et traiter les données !

Une trame de programme est proposée pour les exercices 2 à 4.

## 1 A vous de jouer

### 1.1 Courbes de répartition d'un diacide

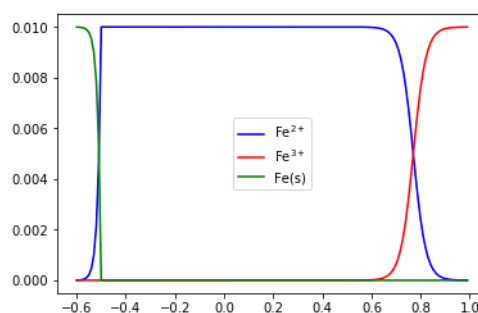
Bon, on commence par un cas simple<sup>1</sup> le tracé des courbes de répartition d'un diacide. Il s'agit donc de tracer le pourcentage du diacide, de l'ampholyte et de la dibase en fonction du pH.

A l'occasion, testez avec vos élèves en leur donnant  $\%(A^{2-}) = \frac{100}{1 + \frac{h}{K_2} + \frac{h^2}{K_1 K_2}}$

### 1.2 To be or not to be !

On complique un peu avec un diagramme d'existence.

L'objectif est de tracer le diagramme suivant représentant l'évolution des quantités de matière (en mole) en fer(s), en ions  $Fe^{2+}$  et  $Fe^{3+}$  présents dans 1,0 litre de solution en fonction du potentiel.



On définit, comme variables globales du système,  $n_0$  (quantité totale de matière (sous toute ses formes) de l'élément fer dans un litre d'eau),  $E_1$  potentiel standard du couple  $Fe^{2+}/Fe(s)$ ,  $E_2$  potentiel standard du couple  $Fe^{3+}/Fe^{2+}$ .

Dans la mesure du possible (sauf si vous n'y arrivez pas autrement !) on ne cherchera pas à déterminer explicitement la valeur de la frontière du domaine d'existence du fer à l'aide d'une approximation de chimiste !

1. Au moins d'un point de vue théorique pour nous !

### 1.3 Courbes de répartition d'un polyacide

L'idée est de pouvoir tracer les courbes de répartition d'un polyacide sans avoir à programmer le monoacide, le diacide, le triacide... en gros, on aimerait bien, par exemple, tracer les courbes de répartition de l'acide phosphorique par le simple appel de `plot_repartition(H3PO4)`.

Forcément, cette variable *H3PO4* doit correspondre à un certain nombre de données, au minimum, les constantes d'acidité. Comment modéliser le système « H3PO4 » ?

On décide, par exemple, de définir un système sous la forme d'une liste

*systeme* = [*nom*, *charge*, *pK1*, *pK2*]<sup>2</sup> :

- *nom* est le nom de la base (sans indiquer la charge : PO4 par exemple) ;
- *charge* est la charge de l'espèce la plus basique
- viennent ensuite les différents *pK<sub>a</sub>* de l'acide

Par exemple, pour l'acide phosphorique on aurait *H3PO4* = ["P04", -3, 2, 7, 12].

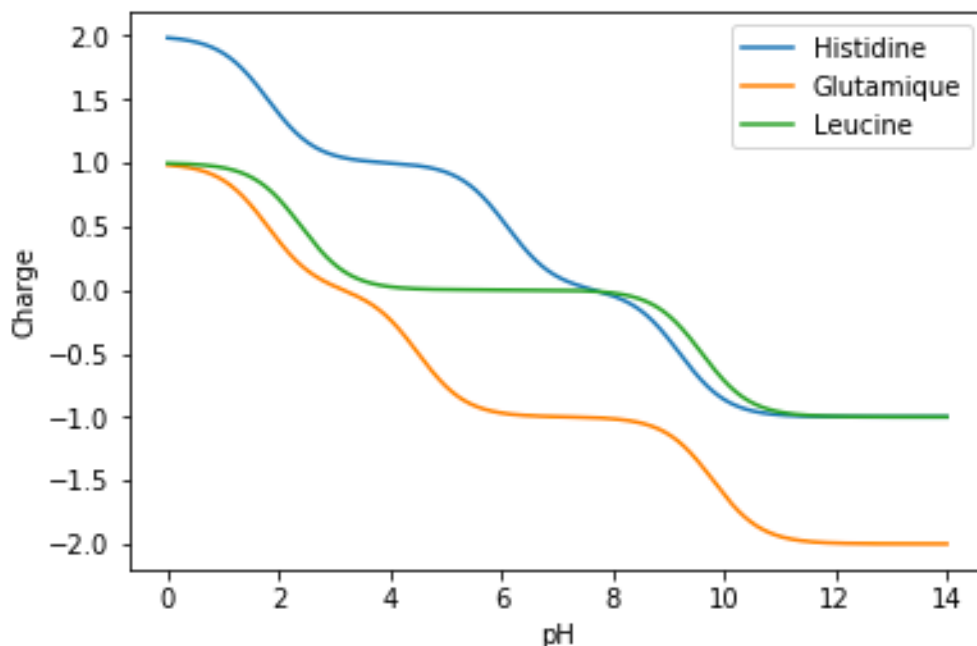
L'appel de `plot_repartition(systeme)` doit alors tracer la courbe de répartition du système étudié quel qu'il soit.

Pour ceux qui sont le plus à l'aise, on générera automatiquement les légendes en codant un éditeur de formule très rudimentaire. Celui-ci nommera les différentes espèces sous la forme : *H2PO4*<sup>-</sup> ou *HPO4*<sup>2-</sup> par exemple.

### 1.4 MLL's Challenge

Alors, en utilisant la trame du programme fournie et quelques fonctions de l'exercice précédent, vous pourrez « aisément :) » tracer la charge des acides aminés proposés dans les données en fonction du pH.

Gare au regard noir de MLL si vous n'y arrivez pas !



1

2. C'est ainsi que l'on pourrait, par exemple, récupérer les informations suite à une requête dans une base de données !

On utilise la même modélisation des données que dans l'exercice précédent. La trame du programme est :

```
2 Histidine = ["Histidine", -1, 1.8, 6.1, 9.2]
3 Glutamique = ["Glutamique", -2, 1.8, 4.5, 9.8]
4 Leucine = ["Leucine", -1, 2.4, 9.6]
5
6
7 def plot_charge(AA, C0):# AA acide aminé étudié C0
   concentration du système
8     PH = []
9     Q = []
10    ...
11    plt.plot(PH, Q, label = AA[0])
12
13 plot_charge(Histidine, 1)
14 plot_charge(Glutamique, 1)
15 plot_charge(Leucine, 1)
16 plt.ylabel("Charge")
17 plt.xlabel("pH")
18 plt.legend()
19 plt.show()
```

L'objectif est donc de compléter le corps de la routine `plot_charge` afin de tracer la courbe charge  $Q$  en fonction du pH  $PH$ .

## 2 Quelques pistes. . .

### 2.1 Courbes de répartition d'un diacide

La solution la plus simple est vraisemblablement de coder les fonctions pourcentages du diacide, de l'ampholyte et de la dibase. . . ou alors, anticiper un peu pour l'exercice 3!

L'appel de ces 3 fonctions dans une boucle `while` sur le `pH` permet alors de créer 4 tableaux de valeurs `PH`, `H2A`, `HA` et `A`.

### 2.2 To be or not to be !

To be or not to be! se dit le fer. . . mais comment savoir s'il y a ou s'il n'y a pas de fer pour une valeur du potentiel donné?

Une première approche consisterait à dire que vu l'écart des potentiels, on n'a pas de Fe(III) à la frontière Fe/Fe(II). On cherche alors le potentiel de cette frontière. Puis on code trace les courbes en faisant un test entre la valeur du potentiel de cette frontière et le potentiel courant.

Peut-être de façon plus élégante, on pourrait définir des fonctions donnant la quantité de Fe(II), Fe(III) et Fe en présence de fer ainsi que la quantité de Fe(II) et Fe(III) en absence de fer. Pour chaque valeur du potentiel, on teste s'il y a ou s'il n'y a pas de fer et on utilise la fonction appropriée.

### 2.3 Courbes de répartition d'un polyacide

Pas simple tout ça. . .

Il y a, certe, une méthode de bourrin. On code les pourcentages des différentes formes dans les différents cas de figure et on fait des tests. Mais bon, on n'est pas là pour ça!

Alors, je vous guide un peu vers une solution possible. . .

En fait, le plus dur est, dans le cas général, de pouvoir calculer, pour un pH et une concentration de référence donnée<sup>3</sup> la concentration de l'espèce la plus basique.

Plutôt qu'écrire une fonction mathématique générale, tentons une récurrence du chimiste! on doit calculer  $\frac{C_0}{1 + 10^{pK_1 - pH}}, \frac{C_0}{1 + 10^{pK_2 - pH} + 10^{pK_2 + pK_1 - 2pH}},$

$\frac{C_0}{1 + 10^{pK_3 - pH} + 10^{pK_3 + pK_2 - 2pH} + 10^{pK_3 + pK_2 + pK_1 - 3pH}}$  etc s'il y a, respectivement, 1, 2, 3 etc acidités.

Diable, un bel exercice de travail sur les listes en perspectives. Vous essaieriez d'encapsuler tout ça dans une fonction `c_base(C0, pH, pKa)` qui accepte comme paramètre la concentration totale `C0`, le pH de la solution `pH` et la liste des différents pKa (dans l'ordre dans la variable `pKa`) et qui retourne la concentration de l'espèce la plus basique. Chez moi cette fonction n'a que 10 lignes.

Dans le début de la fonction `plot_repartition(acide)` il faudra extraire le tableau des pKa nécessaire à la fonction précédente puis. . . yapluka! Je vous conseille quand même de tracer les courbes une par une dans une boucle `for` sur les différentes espèces présentes.

Si vous y arrivez... bravo! et si en prime on a une légende (rudimentaire) pour les courbes, c'est le paradis!

3. On pourrait se passer de cette concentration mais on en a besoin dans l'exercice suivant et dans d'autres plus tard!

**2.4****MLL's Challenge**

Bon, le plus dur est passé.

Si vous n'avez pas trouvé la réponse à l'exercice précédent, copiez le code de la fonction `c_base(CO, pH, pKa)`, vous en aurez besoin.

Avec le code de cette fonction, le calcul devrait être plus facile!

### 3 Solutions...

#### 3.1 Courbes de répartition d'un diacide

Pas de soucis, j'espère dans le code de cet exercice.

```

1 def pct_H2A(pH, pK1, pK2):
2     return 100/(1 + 10**(pH-pK1)+10**(2*pH-pK1-pK2))
3
4 def pct_HA(pH, pK1, pK2):
5     return 100/(10**(pK1-pH) + 1 + 10**(pH-pK2))
6
7 def pct_A(pH, pK1, pK2):
8     return 100/(10**(pK1 + pK2 - 2*pH) + 10**(pK2-pH) + 1 )
9
10 H2A = []
11 HA = []
12 A = []
13 PH = []
14 pKa1 = 4
15 pKa2 = 8
16 pH = 0
17 while pH < 14 :
18     PH.append(pH)
19     H2A.append(pct_H2A(pH, pKa1, pKa2))
20     HA.append(pct_HA(pH, pKa1, pKa2))
21     A.append(pct_A(pH, pKa1, pKa2))
22     pH = pH + 0.01
23
24 plt.plot(PH, H2A, 'r', label = "H2A")
25 plt.plot(PH, HA, 'g', label = "HA-")
26 plt.plot(PH, A, 'b', label = "A2-")
27 plt.show()

```

On pourrait, tout aussi bien avoir des fonctions mettant en jeu  $h$ ,  $K1$  et  $K2$ .

Une prime à ceux qui n'utilisent qu'une seule fonction : la base par exemple et qui calculent de proche en proche le pourcentage de l'ampholyte puis du diacide.

#### 3.2 To be or not to be !

```

1 NO = 0.01
2 E1 = - 0.44
3 E2 = 0.77
4
5 def nFe2(E):
6     return NO/(1+10**((E-E2)/0.06))
7 def nFe3(E):
8     return NO/(1+10**(-(E-E2)/0.06))
9 def nFe2_Fe(E):
10    return 10**((E-E1)/0.03)
11 def nFe3_Fe(E):
12    return nFe2_Fe(E)*10**((E-E2)/0.06)
13 def nFe(E):
14    return NO - (nFe2_Fe(E)+nFe3_Fe(E))

```

```

15
16 E=[]
17 FE=[]
18 FE2=[]
19 FE3=[]
20
21 pot = -0.6
22 while pot < 1 :
23     E.append(pot)
24     if nFe(pot)>=0:
25         FE.append(nFe(pot))
26         FE2.append(nFe2_Fe(pot))
27         FE3.append(nFe3_Fe(pot))
28     else :
29         FE.append(0)
30         FE2.append(nFe2(pot))
31         FE3.append(nFe3(pot))
32     pot = pot + 0.01
33
34 plt.plot(E, FE2, 'b', label="Fe2+")
35 plt.plot(E, FE3, 'r', label="Fe3+")
36 plt.plot(E, FE, 'g', label="Fe(s)")
37 plt.ylabel("Quantité de matière")
38 plt.xlabel("E(V)")
39 plt.legend()
40 plt.show()

```

Quelques commentaires sur ce programme :

- Lignes 5 à 8 : on définit les fonctions donnant les quantités de matière en absence de fer.
- En présence de fer : la loi de NERNST permet de déterminer la quantité de  $\text{Fe}^{2+}$  en présence de fer, d'en déduire celle de  $\text{Fe}^{3+}$ . On calcule ligne 14 la quantité de fer par différence entre la quantité de référence  $N_0$  et les deux quantités précédentes.
- On remplit classiquement les tableaux de valeur à l'aide d'une boucle `while` sur le potentiel. La clé de la résolution de l'exercice est l'appel de la fonction `nFe(E)` qui permet de savoir s'il y a ou s'il n'y a pas de fer présent et donc de choisir les bonnes fonctions pour la suite.

### 3.3 Courbes de répartition d'un polyacide

```

1 monoacide = ["A", -1, 4]
2 diacide = ["A", -2, 4, 8]
3 triacide = ["A", -3, 2, 7, 12]
4
5
6 def c_base(C0, pH, pKa):
7     nb_acide = len(pKa)
8     denum = 1
9     for i in range(1, nb_acide+1):
10         s = 0
11         for j in range(len(pKa)-1, len(pKa)-1-i, -1) :
12             s = s + pKa[j]
13         denum = denum + 10**(s - i*pH)
14     c = C0/denum
15     return c

```

```

16
17 def nom(nbH, nomBase, chargeBase):
18     name = ""
19     if nbH > 0 :
20         name = "H"
21         if nbH >= 2 :
22             name = name + str(nbH)
23     name = name + nomBase
24     charge = chargeBase + nbH
25     if charge == -1 :
26         name = name + "-"
27     elif charge == 1 :
28         name = name + "+"
29     elif charge < 0 :
30         name = name + str(abs(charge)) + "-"
31     elif charge > 0 :
32         name = name + str(charge) + "+"
33     return name
34
35
36 def plot_repartition(acide):
37     pKa = [acide[i] for i in range(2, len(acide))]
38     nb_acide = len(acide) - 2
39     for i in range (nb_acide + 1):
40         PH = []
41         PCT = []
42         pH = 0
43         while pH <= 14 :
44             PH.append(pH)
45             c = c_base(100, pH, pKa)
46             for j in range(i) :
47                 c = c * 10**(pKa[len(pKa)-j-1]-pH)
48             PCT.append(c)
49             pH = pH + 0.01
50         plt.plot(PH, PCT, label = nom(i, acide[0], acide[1]))
51
52 plot_repartition(triacide)
53 plt.ylabel("Pourcentage")
54 plt.xlabel("pH")
55 plt.legend()
56 plt.show()

```

Dans la fonction `plot_repartition` :

- On laisse tomber les deux premières grandeurs stockées dans la liste passée en paramètre et on stocke les suivantes dans la variable *pKa*.
- La longueur de cette liste (ou celle d'origine - 2) correspond au nombre d'acides.
- Le plus simple des de tracer les courbes de répartition une par une, en partant de l'espèce la plus basique<sup>4</sup>.
- On a, bien sûr, *nb\_acide* + 1 courbes à tracer...

---

4. Cette solution est loin d'être la plus performante, il vaudrait mieux pour un pH donné calculer toutes les concentrations de proche en proche en partant de la plus basique et de les stocker. Dans ces conditions, il faudrait utiliser des listes de liste et non pas une liste unique comme ici.



- Les lignes les plus difficiles à coder sont les lignes 46 et surtout 47 pour passer de la concentration de la base aux différents acides. Il faudra peut-être tracer quelques courbes aux allures surprenantes avant d'arriver au résultat... une solution empirique permet parfois d'arriver plus vite aux résultats qu'une prise de tête avec une formule mathématique mais, je ne suis pas matheux!

Passons à cette fameuse fonction `c_base`... là encore, j'avoue qu'un peu d'essai-erreur peu être une solution!

- Bien sûr, le nombre d'acide correspond à la longueur du tableau *pKa*.
- Pour la *i*ème acidité, on a, au dénominateur,  $1 + i$  termes; chacun de ces termes est une puissance de 10.
- Ces valeurs de pKa doivent être lus de la fin vers le début pour obtenir la somme des termes dans la puissance de 10.

Reste éventuellement à écrire le nom de l'espèce pour la légende. On fait une concaténation de chaîne (on verra une activité bientôt sur les chaînes de caractères) avec H puis le nombre de H (sauf si c'est 1) puis le nom de la base sans la charge puis la charge calculée.

### 3.4 MLL's Challenge

```

1 Histidine = ["Histidine", -1, 1.8, 6.1, 9.2]
2 Glutamique = ["Glutamique", -2, 1.8, 4.5, 9.8]
3 Leucine = ["Leucine", -1, 2.4, 9.6]
4
5 def plot_charge(acide, C0):
6     pKa = [acide[i] for i in range(2, len(acide))]
7     nb_acide = len(acide) - 2
8     PH = []
9     Q = []
10    pH = 0
11    while pH <= 14:
12        PH.append(pH)
13        q = 0
14        for i in range(nb_acide + 1):
15            c = c_base(C0, pH, pKa)
16            for j in range(i):
17                c = c * 10**(pKa[len(pKa)-j-1]-pH)
18            q = q + c*(acide[1]+i)
19        Q.append(q)
20        pH = pH + 0.01
21    plt.plot(PH, Q, label = acide[0])
22
23 plot_charge(Histidine, 1)
24 plot_charge(Glutamique, 1)
25 plot_charge(Leucine, 1)
26 plt.ylabel("Charge")
27 plt.xlabel("pH")
28 plt.legend()

```

Ce code ressemble relativement au précédent.

Il faut toutefois inverser les boucles sur le pH et sur les espèces car, pour un pH donné (ligne 12), il faut parcourir toutes les espèces (ligne 15) afin de connaître leur concentration (ligne 16 à 18) puis leur contribution à la charge (ligne 19).