Capacité numérique

Composition chimique du système dans l'état final :

Déterminer, à l'aide d'un langage de programmation, l'état final d'un système à l'équilibre, siège d'une transformation, modélisée par une ou deux réactions à partir des conditions initiales et valeur(s) de la(es) constante(s) thermodynamique(s) d'équilibre.

Incertitudes-types composées :

Simuler, à l'aide d'un langage de programmation ou d'un tableur, un processus aléatoire de type Monte Carlo permettant de caractériser la variabilité de la valeur d'une grandeur composée.

Régression linéaire :

À l'aide d'un langage de programmation ou d'un tableur, simuler un processus aléatoire de variation des valeurs expérimentales de l'une des grandeurs – simulation Monte-Carlo – pour évaluer l'incertitude type sur les paramètres du modèle.

Cinétique en réacteur fermé de composition uniforme :

À l'aide d'un langage de programmation ou d'un logiciel dédié, et à partir de données expérimentales, tracer l'évolution temporelle d'une concentration, d'une vitesse volumique de formation ou de consommation, d'une vitesse de réaction et tester une loi de vitesse donnée.

Modélisation microscopique d'une transformation chimique :

- Établir un système d'équations différentielles et le résoudre numériquement afin de visualiser l'évolution temporelle des concentrations et de leurs dérivées dans le cas d'un mécanisme à deux actes élémentaires successifs. Mettre en évidence l'étape cinétiquement déterminante ou l'approximation de l'état quasi-stationnaire d'un intermédiaire réactionnel.

- Établir un système d'équations différentielles et le résoudre numériquement, avec un langage de programmation, afin de visualiser l'évolution des concentrations au cours du temps pour mettre en évidence les situations de contrôle cinétique ou thermodynamique.

Réactions acide-base et de précipitation :

Tracer, à l'aide d'un langage de programmation, le diagramme de distribution des espèces d'un ou plusieurs couples acide-base, ou d'espèces impliquées dans une réaction de précipitation.

Annexe 3 : outils numériques

La prise en compte de capacités de codage en langage Python incluant l'utilisation de fonctions extraites de diverses bibliothèques dans la formation des étudiants vise à une meilleure appréhension des principes mis en œuvre par les différents logiciels de traitement des données dont l'utilisation est par ailleurs toujours recommandée et à mobiliser ces capacités dans un contexte concret, celui de la physique-chimie. Cette formation par le codage permet également de développer des capacités utiles à la physique-chimie comme le raisonnement, la logique ou la décomposition d'un problème complexe en étapes plus simples.

Le tableau ci-dessous explicite ces outils ainsi que les capacités exigibles en fin de première année. Il sera complété dans le programme de seconde année.

Outils numériques	Capacités exigibles
1. Outils graphiques	
Représentation graphique d'un nuage de points.	Utiliser les fonctions de base de la bibliothèque **matplotlib** pour représenter un nuage de points et rendre le graphe exploitable (présence d'une légende, choix des échelles…).
Représentation graphique d'une fonction.	Utiliser les fonctions de base de la bibliothèque **matplotlib** pour tracer la courbe représentative d'une fonction et rendre le graphe exploitable (présence d'une légende, choix des échelles…).
2. Équations algébriques	
Résolution d'une équation algébrique ou d'une équation transcendante : – méthode dichotomique, – méthode de Newton.	Déterminer, en s'appuyant sur une représentation graphique, un intervalle adapté à la recherche numérique d'une racine par la méthode dichotomique ou par la méthode de Newton. Mettre en œuvre la méthode dichotomique ou la méthode de Newton afin de résoudre une équation avec une précision donnée. Utiliser les fonctions **bisect** ou **newton** de la bibliothèque **scipy.optimize** (leurs spécifications étant fournies).
Systèmes linéaires de n équations indépendantes à n inconnues.	Définir les matrices A et B adaptées à la représentation matricielle AX = B du système à résoudre. Utiliser la fonction **solve** de la bibliothèque **numpy.linalg** (sa spécification étant fournie).
3. Intégration – Dérivation	
Calcul approché du nombre dérivé d'une fonction en un point.	Utiliser un schéma numérique centré ou décentré pour déterminer une valeur approchée du nombre dérivé d'une fonction en un point.
4. Équations différentielles	
Équations différentielles d'ordre 1.	Mettre en œuvre la méthode d'Euler explicite afin de résoudre une équation différentielle d'ordre 1 ou un système d'équations différentielles.
5. Statistiques	
Régression linéaire.	Utiliser la fonction **polyfit** de la bibliothèque numpy (sa spécification étant fournie) pour exploiter des données. Utiliser la fonction **random.normal** de la bibliothèque numpy (sa spécification étant fournie) pour simuler un processus aléatoire.