Objectifs de formation et programme d'informatique de la classe préparatoire scientifique ATS métiers de la chimie

I Objectifs de formation

1 Généralités

L'informatique, omniprésente dans les différentes sphères de l'entreprise, de la recherche, des services, de la culture et des loisirs, repose sur des mécanismes fondamentaux devant être maîtrisés par les futurs ingénieurs, enseignants et chercheurs qui auront à s'en servir pour agir en connaissance de cause dans leur vie professionnelle.
La rapide évolution des outils informatiques et des sciences du numérique dans tous les secteurs de l'ingénierie (industrielle, logicielle et des services) et de la recherche rend indispensable un enseignement de l'informatique spécifiquement conçu pour l'étudiant de CPGE scientifiques. Celui-ci devra pouvoir dans sa vie professionnelle communiquer avec les informaticiens de son entreprise ou de son laboratoire, participer aux prises de décision en matière de systèmes d'information, posséder des connaissances de base nécessaires à la compréhension des défaillances et des risques informatiques, ainsi que des solutions permettant d'y remédier, et exploiter à bon escient les résultats de calculs numériques. Pour ce faire, il devra comprendre des concepts tels que la précision numérique, la faisabilité, l'efficacité, la qualité et les limites de solutions informatiques, ce qui requiert une certaine familiarité avec les architectures matérielles et logicielles, les systèmes d'exploitation, le stockage des données. Cette diversité d'exigences impose une formation à la fois fondamentale et appliquée.
Au niveau fondamental, on se fixe pour objectif la maîtrise d'un certain nombre de concepts de base, et avant tout, la conception rigoureuse d'algorithmes et le choix de représentations appropriées des données. Ceci impose une expérience pratique de la programmation et de la manipulation informatique de données, notamment d'origine expérimentale ou industrielle, et parfois disponibles en ligne.
Au niveau des applications, la rapidité d'évolution des technologies logicielles et matérielles renforce l'intérêt de présenter des concepts fondamentaux pérennes sans s'attacher outre mesure à la description de technologies, protocoles ou normes actuels. En revanche, la formation s'attachera à contextualiser le plus souvent possible les activités pratiques en s'appuyant sur les autres disciplines scientifiques : chimie, physique, mathématiques, génie des procédés.

2 Compétences visées

Cet enseignement doit permettre de développer les compétences suivantes :

Analyser et modéliser	un problème, une situation ;
Imaginer et concevoir	une solution algorithmique modulaire, utilisant des méthodes de programmation appropriées pour le problème étudié ;
Traduire	un algorithme dans un langage de programmation moderne et généraliste ;
Spécifier	rigoureusement les modules ou fonctions ;
Évaluer, contrôler, valider	des algorithmes et des programmes ;
Communiquer	à l'écrit ou à l'oral, une problématique, une solution ou un algorithme, une documentation.

L'étude et la maîtrise de quelques algorithmes fondamentaux, associés à l'apprentissage de la syntaxe du langage de programmation choisi, permettent de développer des méthodes (ou paradigmes) de programmation appropriés, fiables et efficaces : programmation impérative, approche descendante, programmation structurée, sensibilisation au coût en temps et en mémoire, documentation des programmes en vue de leur réutilisation et possibles modifications ultérieures.
La pratique régulière de la résolution de problèmes par une approche algorithmique et des activités de programmation qui en résultent constitue un aspect essentiel de l'apprentissage de l'informatique. Il est éminemment souhaitable que les exemples choisis ainsi que certains exercices d'application soient directement inspirés par les enseignements de physique et chimie, de mathématiques, et de génie des procédés.

II Programme

1. Introduction : présentation du système informatique utilisé et éléments d'architecture des ordinateurs

Une séance introductive sera consacrée à présenter et à familiariser les étudiants :
- aux principaux composants d'une machine numérique telle que l'ordinateur personnel, une tablette, etc. : sources d'énergie, mémoire vive, mémoire de masse, unité centrale, périphériques d'entrée-sortie, ports de communication avec d'autres composants numériques (aucune connaissance particulière des composants cités n'est cependant exigible) ;
- à la manipulation d'un système d'exploitation (gestion des ressources, essentiellement : organisation des fichiers, arborescence, droits d'accès, de modification, entrées/sorties) ;
- à la manipulation d'un environnement de développement.

La principale capacité développée dans cette partie de la formation est :
- manipuler en mode « utilisateur » les principales fonctions d'un système d'exploitation et d'un environnement de développement.

2. Algorithmique et programmation

2.a Outils employés

L'enseignement s'appuie sur un environnement logiciel complet, dédié aussi bien à la programmation qu'à des tâches de calcul scientifique en lien avec l'étude des problèmes de simulation. Au moment de la conception de ce programme, l'environnement sélectionné est Scilab.
Afin d'en permettre rapidement une utilisation dans d'autres enseignements, une séance de présentation de cet environnement sera prévue en début d'année. Les travaux pratiques conduiront à éditer et manipuler fréquemment des codes sources et des fichiers. Les étudiants doivent être familiarisés, au sein de l'environnement logiciel, avec les tâches de création d'un fichier source, d'édition d'un programme, de gestion des fichiers, d'exécution et d'arrêt forcé d'un programme.
L'étude approfondie de l'environnement logiciel n'est pas une fin en soi et n'est pas un attendu du programme.

Des textes réglementaires ultérieurs pourront mettre à jour ce choix d'environnement en fonction des évolutions et des besoins.

2.b Algorithmique

Les compétences en matière d'algorithmique et de programmation étant profondément liées, il est souhaitable que ces deux sujets soient abordés de concert, même si pour des raisons de clarté d'exposition ils sont ici séparés.
L'introduction à l'algorithmique contribue à apprendre à l'étudiant à analyser, à spécifier et à modéliser de manière rigoureuse une situation ou un problème. Cette démarche algorithmique procède par décomposition en sous-problèmes et par affinements successifs. L'accent étant porté sur le développement raisonné d'algorithmes, leur implantation dans un langage de programmation n'intervient qu'après une présentation organisée de la solution algorithmique, indépendante du langage choisi.
Les invariants de boucles sont introduits pour s'assurer de la correction des segments itératifs.
La notion de coût d'un algorithme en temps et en mémoire est introduite sur des exemples simples.
Pour faire mieux comprendre la notion d'algorithme et sa portée universelle, on s'appuie sur un petit nombre d'algorithmes simples, classiques et d'usage universel, que les étudiants doivent savoir expliquer et programmer, voire modifier selon les besoins et contraintes des problèmes étudiés.

Contenus	Précisions et commentaires
Recherche dans une liste, recherche du maximum dans une liste de nombres, calcul de la moyenne et de la variance.	La notion de coût en nombre d'opérations pourra être abordée ici : nombre de comparaisons pour la recherche du maximum ; nombre d'additions pour le calcul de la moyenne.
Recherche par dichotomie dans un tableau trié. Recherche par dichotomie du zéro d'une fonction continue et monotone.	La question de la précision du calcul pourra être abordée ici, de même que les phénomènes de dépassement de capacité (ou « overflow ») conduisant à des résultats faux ou à des erreurs d'arrondis. Le problème de la comparaison avec 0 sera mis en avant.
Méthodes des rectangles et des trapèzes pour le calcul approché d'une intégrale sur un segment.	La question de la précision du calcul pourra à nouveau être abordée, en particulier par une comparaison des deux méthodes. On pourra également faire la comparaison avec les outils d'intégration numérique fournis par l'environnement logiciel.

Les principales capacités développées dans cette partie de la formation sont :
- comprendre un algorithme et expliquer ce qu'il fait ;
- modifier un algorithme existant pour obtenir un résultat différent ;
- concevoir un algorithme répondant à un problème précisément posé ;
- expliquer le fonctionnement d'un algorithme ;
- écrire des instructions conditionnelles avec alternatives, éventuellement imbriquées ;
- justifier qu'une itération (ou boucle) produit l'effet attendu au moyen d'un invariant ;
- démontrer qu'une boucle se termine effectivement ;
- s'interroger sur le coût en temps d'un algorithme.

Les étudiants devront être capables de programmer au sein de l'environnement logiciel indiqué ci-dessus les différents algorithmes étudiés.

2.c Programmation

On insistera sur une organisation modulaire des programmes ainsi que sur la nécessité d'une programmation structurée et parfaitement documentée.

Contenus	Précisions et commentaires
Variables : notion de type et de valeur d'une variable, types simples.	Les types simples présentés sont les entiers, flottants, booléens et chaînes de caractères.
Expressions et instructions simples : affectation, opérateurs usuels, distinction entre expression et instruction.	Les expressions considérées sont à valeurs numériques, booléennes ou de type chaîne de caractères.
Instructions conditionnelles : expressions booléennes et opérateurs logiques simples, instruction if. Variantes avec alternative (else).	Les étudiants devront être capables de structurer et comprendre plusieurs niveaux d'alternatives implantées par des instructions conditionnelles imbriquées.
Instructions itératives : boucles for, boucles conditionnelles while.	Les sorties de boucle (instruction break) peuvent être présentées et se justifient uniquement lorsqu'elles contribuent à simplifier notablement la programmation sans réelle perte de lisibilité des conditions d'arrêt.
Fonctions : notion de fonction (au sens informatique), définition dans le langage utilisé, paramètres (ou arguments) et résultats, portée des variables.	On distingue les variables locales des variables globales et on décourage l'utilisation des variables globales autant que possible.
Manipulation de quelques structures de données : chaînes de caractères (création, accès à un caractère, concaténation), listes (création, ajout d'un élément, suppression d'un élément, accès à un élément, extraction d'une partie de liste), tableaux à une ou plusieurs dimensions.	On met en évidence le fait que certaines opérations d'apparence simple cachent un important travail pour le processeur.
Fichiers : notion de chemin d'accès, lecture et écriture de données numériques ou de type chaîne de caractères depuis ou vers un fichier.	On encourage l'utilisation de fichiers en tant que supports de données ou de résultats avant divers traitements, par exemple graphiques.

Les exemples de programmation ne se limitent pas à la traduction des algorithmes introduits en partie 2.b.
Les principales capacités développées dans cette partie sont les suivantes :
- concevoir l'en-tête (ou la spécification) d'une fonction, puis la fonction elle-même ;

- traduire un algorithme dans un langage de programmation ;
- gérer efficacement un ensemble de fichiers correspondant à des versions successives d'un fichier source ;
- rechercher une information au sein d'une documentation en ligne, analyser des exemples fournis dans cette documentation ;
- documenter une fonction, un programme plus complexe.

3. Ingénierie numérique et simulation

3.a Objectifs et organisation de cet enseignement

Dans cette partie de programme, on étudie le développement d'algorithmes numériques sur des problèmes scientifiques étudiés et mis en équation dans les autres disciplines. La pédagogie par projets est encouragée.

3.b Outils employés

L'objectif est de familiariser les étudiants avec un environnement de simulation numérique. Cet environnement doit permettre d'utiliser des outils de calcul numérique et leur documentation pour développer et exécuter des programmes numériques. On veillera à faire aussi programmer par les étudiants certains des algorithmes étudiés. Aucune connaissance des fonctions fournies par l'environnement n'est exigible des étudiants. Au moment de l'élaboration de ces programmes d'enseignement, l'atelier logiciel Scilab est l'environnement choisi.

3.c Simulation numérique

Il s'agit d'apprendre aux étudiants à utiliser des algorithmes numériques simples et/ou à utiliser des outils existants pour résoudre des problèmes étudiés et mis en équation dans les autres disciplines. Le problème d'origine doit être exposé mais la modélisation (et la mise en équations) n'est pas un objectif de ce programme.
Dans cette partie, on n'aborde pas les aspects théoriques qui relèvent des autres enseignements scientifiques. Seules la mise en œuvre constructive des algorithmes et l'analyse empirique des résultats sont concernées. On s'attache à comparer la solution numérique à une solution analytique quand elle existe, à des résultats expérimentaux, aux solutions obtenues en utilisant les fonctions de l'environnement de travail choisi. On illustre ainsi les performances de différents algorithmes pour la résolution des problèmes. On met l'accent sur les aspects pratiques comme l'impact des erreurs d'arrondi sur les résultats, les conditions d'arrêt, le coût en temps de calcul ou le stockage en mémoire.

Contenus	Précisions et commentaires
Outils logiciels existants : utilisation de quelques fonctions de l'environnement logiciel et de leur documentation en ligne.	On met en évidence l'intérêt de faire appel aux outils numériques déjà existants au sein de l'environnement, évitant de devoir réinventer des solutions à des problèmes bien connus. La recherche des spécifications des fonctions concernées joue un rôle essentiel pour le développement de solutions fiables aux problèmes posés.
Problème stationnaire à une dimension, linéaire ou non conduisant à la résolution approchée d'une équation numérique. Méthode de dichotomie, méthode de Newton.	On souligne les différences du comportement informatique des deux algorithmes en termes de rapidité. On illustre à nouveau le problème du test d'arrêt (inadéquation de la comparaison à zéro).
Problème dynamique à une dimension, linéaire ou non, conduisant à la résolution approchée d'une équation différentielle ordinaire par la méthode d'Euler.	On compare les résultats obtenus avec les fonctions de résolution approchée fournies par l'environnement logiciel. On met en évidence l'impact du pas de discrétisation et du nombre d'itérations sur la qualité des résultats et sur le temps de calcul.
Problème discret multidimensionnel, linéaire, conduisant à la résolution d'un système linéaire inversible (ou de Cramer) par la méthode de Gauss avec recherche partielle du pivot.	La méthode de Gauss étant introduite dans le cours de mathématiques, il est nécessaire de se coordonner avec le professeur de mathématiques pour traiter cette question. Il ne s'agit pas de présenter cet algorithme mais de l'exécuter pour étudier sa mise en œuvre et les problèmes que pose cette démarche. On illustre l'impact de la taille des matrices sur le temps de calcul.

Les principales capacités développées dans cette partie de la formation sont :
- réaliser un programme complet structuré allant de la prise en compte de données expérimentales à la mise en forme des résultats permettant de résoudre un problème scientifique donné ;
- étudier l'effet d'une variation des paramètres sur le temps de calcul, sur la précision des résultats, sur la forme des solutions pour des programmes d'ingénierie numérique choisis ;
- utiliser les fonctions de l'environnement logiciel pour résoudre un problème scientifique mis en équation lors des enseignements de chimie, physique, mathématiques, génie des procédés ;
- utiliser les fonctions de l'environnement logiciel pour afficher les résultats sous forme graphique ;
- tenir compte des aspects pratiques comme l'impact des erreurs d'arrondi sur les résultats, le temps de calcul ou le stockage en mémoire.

4. Réalisation d'un projet transdisciplinaire

L'acquisition durable de compétences, même modestes, en informatique repose sur une régularité d'exercices pratiques et s'accorde au mieux avec le développement de projets. Il est donc recommandé que les étudiants mettent en œuvre leurs connaissances en informatique dans le cadre de la réalisation de leur projet transdisciplinaire (alliant chimie, et/ou physique et/ou mathématiques et/ou génie des procédés).
Les thèmes des projets doivent être choisis de manière à représenter la diversité des applications possibles, notamment en physique, chimie et génie des procédés. Le choix des thèmes peut être laissé à la discrétion des étudiants sous réserve d'une supervision de la part des enseignants. Ces derniers veilleront ainsi à ce qu'un volet informatique soit développé au sein de chaque sujet.
Ces projets doivent pouvoir être présentés (sous forme écrite et orale) par les étudiants en mettant en valeur :
- la nature et l'intérêt du problème scientifique étudié ;
- l'approche choisie pour résoudre le problème ;
- l'organisation choisie pour la conduite du projet (répartition des tâches, échéancier) ;

- la structuration de la solution (découpage en diverses tâches et modules) ;
- l'adéquation de la solution par rapport au problème initialement posé.

Capacités intervenant dans cette partie :

- recueillir des informations et mobiliser des ressources ;
- créer des perspectives nouvelles ;
- organiser un travail impliquant un développement logiciel ;
- collaborer au sein d'une équipe pour réaliser une tâche ;
- développer un regard critique sur les résultats obtenus ;
- présenter une solution à l'écrit, à l'oral.