Objectifs de formation et programme d'informatique de la classe préparatoire scientifique d'ATS ingénierie industrielle

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 et les réseaux. 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 : physique, mathématiques, sciences industrielles de l'ingénieur.

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, des structures de données 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, l'utilisation de structures de données adaptées et 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, utilisation de bibliothèques logicielles, 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, de mathématiques, et de sciences industrielles de l'ingénieur.

3. Outil employé

L'enseignement se fonde sur un outil de programmation (langage et bibliothèques) basé sur un langage interprété largement répandu et à source libre. Au moment de la conception de ce programme, l'outil sélectionné est Scilab. Les travaux pratiques conduiront à éditer et manipuler fréquemment des codes sources et des fichiers. Les étudiants doivent être familiarisés 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. Cet outil de calcul scientifique est utilisé en lien avec l'étude des problèmes de simulation. Son étude 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 en fonction des évolutions et des besoins.

II - Programme

1. Architectures matérielles

a) Présentation du système informatique utilisé et éléments d'architecture des ordinateurs

Une ou deux séances introductives seront consacrées 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 de manipuler en mode « utilisateur » les principales fonctions d'un système d'exploitation et d'un environnement de développement.

b) Représentation des nombres et conséquences

Il s'agit de familiariser les étudiants avec les problèmes liés à la représentation finie des nombres et à la discrétisation des modèles numériques. Les calculatrices peuvent servir de support d'étude de ces questions.

Contenus	Précisions et commentaires
Principe de la représentation des nombres entiers en mémoire.	On introduit ou rappelle brièvement le principe des représentations binaire et hexadécimale ainsi que leurs limites.

Principe de la représentation des nombres réels en mémoire.	On se limite à la définition de l'écriture en virgule flottante normalisée et on explique le codage d'un nombre réel en général sans entrer dans les cas particuliers comme les non-nombres « not a number » ou les infinis.
Conséquences de la représentation limitée des nombres réels en machine.	On illustre, sur des exemples simples, pouvant être illustrés au moyen d'une calculatrice, les phénomènes de dépassement de capacité (ou « overflow ») de séquences de calculs conduisant à des résultats faux et erreurs d'arrondis. On illustre aussi le problème de la comparaison à zéro, par exemple dans une équation du second degré.

Les principales capacités développées dans cette partie de la formation sont :
- appréhender les limitations intrinsèques à la manipulation informatique des nombres ;
- initier un sens critique au sujet de la qualité et de la précision des résultats de calculs numériques sur ordinateur.

2. Algorithmique et programmation

a) 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.
La notion de complexité temporelle des algorithmes est introduite de manière expérimentale 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.	
Recherche par dichotomie dans un tableau trié. Recherche par dichotomie du zéro d'une fonction continue et monotone.	Les questions de précision du calcul sont en lien avec la partie 1.b.
Méthodes des rectangles et des trapèzes pour le calcul approché d'une intégrale sur un segment.	Les questions de précision du calcul sont en lien avec la partie 1.b.

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 et le documenter ;
- expliquer le fonctionnement d'un algorithme ;
- écrire des instructions conditionnelles avec alternatives, éventuellement imbriquées ;
- s'interroger sur l'efficacité algorithmique temporelle d'un algorithme.

b) 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. On met à profit la structure de tableau d'entiers à deux dimensions pour introduire la notion d'image ponctuelle (« bitmap »). Les algorithmes de traitement d'image seront abordés plus tard.

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 :
- choisir un type de données en fonction d'un problème à résoudre ;
- 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 ;
- modifier une programmation en vue de changer le comportement de tout ou partie d'un système complexe.

3. Ingénierie numérique et simulation

Dans cette partie de programme, on étudie le développement d'algorithmes numériques sur des problèmes scientifiques étudiés et mis en équation. La pédagogie par projets est encouragée.
Il s'agit d'apprendre aux étudiants à utiliser des algorithmes numériques simples et/ou à utiliser des bibliothèques pour résoudre des problèmes étudiés et mis en équation dans les autres disciplines.
Dans cette partie, on n'aborde pas les aspects théoriques des algorithmes étudiés (qui peuvent être traités dans d'autres disciplines). Seules la mise en œuvre constructive des algorithmes et l'analyse empirique des résultats sont concernées. 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, la complexité en temps de calcul.

Contenus	Précisions et commentaires
Bibliothèques logicielles : utilisation de quelques fonctions d'une bibliothèque et de leur documentation en ligne.	On met en évidence l'intérêt de faire appel aux bibliothèques, évitant de devoir réinventer des solutions à des problèmes bien connus. La recherche des spécifications des bibliothèques 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 algébrique ou transcendante. Méthode de dichotomie, méthode de Newton.	On souligne les différences du comportement informatique des deux algorithmes en termes de rapidité. On illustre 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 une bibliothèque numérique. 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.	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 souligne la complexité de l'algorithme en fonction de la taille des matrices et son impact 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 ;
- utiliser les bibliothèques de calcul standard pour résoudre un problème scientifique mis en équation lors des enseignements, de physique, mathématiques, sciences industrielles de l'ingénieur ;
- 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

L'acquisition durable de compétences en informatique repose sur une pratique régulière et s'accorde au mieux avec le développement de projets. Il est donc recommandé de faire réaliser aux étudiants un projet mettant en œuvre les compétences des programmes de la filière.
Pour la réalisation de ce projet, les étudiants peuvent travailler en groupe de taille réduite. Le temps passé sur les projets doit rester modeste. Les thèmes des projets doivent être choisis de manière à représenter la diversité des applications possibles, notamment en physique et sciences industrielles de l'ingénieur.
Les principales capacités développées dans cette partie de la formation sont :
- recueillir des informations et mobiliser des ressources ;
- décomposer un problème complexe en tâches élémentaires ;
- organiser un travail impliquant un développement logiciel ;
- collaborer au sein d'une équipe pour réaliser une tâche ;
- avoir un regard critique sur les résultats obtenus ;
- présenter une solution à l'écrit, à l'oral.

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 ;
- l'adéquation de la solution par rapport au problème initialement posé.