Informatique
en CPGE

L'ingénieur doit maîtriser les concepts fondamentaux de l'informatique pour :

- communiquer avec les informaticiens
- comprendre les questions de complexité algorithmique, de précision numérique, de sécurité
- disposer d'une représentation claire de l'architecture d'un ordinateur, d'un OS, d'un réseau
- savoir écrire du code et utiliser les bibliothèques logicielles
- interroger une base de données
- prendre des décisions stratégiques de développement de son entreprise

Contexte

- Un monde numérique qui connaît une explosion de la masse des données

En conséquence :

- savoir exploiter les résultats de calculs numériques
- manipulation de données d'origine expérimentale ou industrielle, ou disponibles en ligne

• Contexte *(suite)*

- Une obsolescence rapide des technologies

  En conséquence :

  - présenter des *concepts fondamentaux*
  - ne pas se lier à des normes ou protocoles *non pérennes*
  - faire le lien avec les autres disciplines, pour *contextualiser les activités pratiques*

Des compétences pour l'ingénieur

- analyser et modéliser un problème, une situation
- concevoir une solution modulaire, utilisant les méthodes de programmation et les structures de données appropriées
- traduire un algorithme dans un langage de programmation
- spécifier les composants logiciels développés ou utilisés
- évaluer, contrôler, valider des algorithmes et des programmes
- communiquer à l'écrit et à l'oral une problématique, une solution

Le choix du langage

- Python permet une entrée pédagogiquement adaptée
- syntaxe simple et conforme aux standards
- sémantique claire et bien définie
- nombreuses bibliothèques logicielles pour des exemples inspirés par les disciplines:
  - calcul numérique *(NumPy)*
  - bibliothèque scientifique *(SciPy)*
  - visualisation graphique *(Matplotlib)*

Organisation horaire
première année

- 1 h de cours
- 1 h de TD hebdo = 2 h par quinzaine
- « coût » : 2 h élèves/semaine, et
  3 h professeur/semaine

Têtes de chapitres

- Algorithmique et méthodes de programmation : 60 %

  + architecture des machines
  + représentation des nombres
  + présentation du logiciel SciLab pour qu'il puisse être utilisé en SPC ou SI

- Simulation/ingénierie numérique : 25 %
- Bases de données : 15 %

Méthodes de programmation

- Programmation impérative
- Approche descendante
- Programmation structurée, modulaire
- Utilisation de bibliothèques logicielles
- Spécification des modules
- Notions de complexité en temps, en mémoire
- Utilisation des invariants de boucle

Algorithmique

- recherche dans une liste, maximum, moyenne, variance
- dichotomie : dans un tableau trié, recherche d'un zéro d'une fonction monotone
- méthodes des rectangles et des trapèzes
- recherche (naïve) d'un mot dans une chaîne de caractères

Ingénierie numérique et simulation

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. 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.

- méthodes de Newton et de dichotomie
- méthode d'Euler
- pivot de Gauss

- *comparaison avec les fonctions offertes par les bibliothèques standard et/ou des solutions analytiques*
- *utilisation des bibliothèques de visualisation*
- *questions de précision, de complexité*

Initiation aux bases de données

Perspective applicative, à partir d'exemples, utilisation d'outils interactifs

- vocabulaire : relations, attribut, domaine
- opérateurs ensemblistes
- opérateurs spécifiques de l'algèbre relationnelle (projection, sélection, renommage, jointure…)
- concept client-serveur, architecture trois-tiers

Deuxième année : algorithmique et programmation

- piles
- récursivité
- tris : par insertion, quicksort, tri fusion
- menu à la carte :
  - traitement d'images
  - codage, chiffrage, cryptographie
  - transmission fiable : sommes de contrôle (checksum), codes correcteurs
  - graphes, algorithme de Dijkstra
  - programmation objet : interfaces graphiques