COURS D'ECOLE DOCTORALE 2010
PROGRAMMATION SCIENTIFIQUE

RÉSUMÉ. L'informatique scientifique, même si elle partage beaucoup avec d'autres domaines, possède ses propres contraintes : volonté de calculer vite, grande taille des problèmes, recours quasi obligatoire au parallélisme. Le but de ce cours est : de donner des connaissances de base saines en programmation scientifique, utilisables dans tous les domaines où les mathématiques interviennent (calcul scientifique, mathématiques effectives etc.). Le cours sera articulé autour d'une étude de cas : comment implémenter sur machine la résolution d'un problème précis et non trivial, problème qui servira de fil conducteur pour le cours.

Note : *le problème proviendra de la résolution numérique d'équations aux dérivées partielles, mais aucune connaissance dans ce domaine ne sera nécessaire : les méthodes et concepts seront introduits sans les mathématiques qui les accompagnent, libre à chacun de les approfondir ou non.*

Durée du cours : *24h*

Intervenants : *Thierry Dumont*[1]*, Violaine Louvet*[2]

Présentation du problème type : équations, méthodes numériques, principes de mise en oeuvre : 2h (Cours).

- Système d'équations de Réaction–Diffusion.
- Discrétisation en temps et en espace.
- Algorithmes.
- Les liens avec le reste du cours.

Architecture des ordinateurs, concepts du parallélisme, nouvelles architectures : 2h (Cours).

- Processeurs, mémoires, bus, réseau
- Processeurs spécialisés, multi-coeurs
- Modèles de programmation

Structures de données et algorithmie : 2h (Cours).

- représentation des ensembles de nombres ; indirections (pointeurs).
- structures de données complexes et efficaces : piles, files, listes, arbres, tables etc.
- Algorithmes de tris, d'accès aux éléments.
- représentation des matrices (creuses).
- représentation des graphes.
- complexité.

Programmation orientée objet & généricité : le cas de C++ : 4h (cours+TP).

- Principes de la programmation orientée objet
- Classes, attributs, méthodes, constructeurs et destructeurs
- Héritage, encapsulation, polymorphisme
- Visibilité, surcharge.

– template et programmation générique. Exemples de la STL et de BLITZ.
– Bonnes pratiques de programmation.

Mise en œuvre pratique : élaboration d'un premier programme pour la résolution du problème test en C++.

Utilisation des bibliothèques scientifiques de base : 2h (Cours+TP).
– Les différentes bibliothèques scientifiques
– Leur utilisation

Mise en œuvre pratique : utilisation de certaines de ces bibliothèques dans le programme.

Langage de haut niveau : python : 3h (cours + TP).
– Le langage python
– Intérêt des langages de haut niveau dans la programmation scientifique
– Pré-post traitement, visualisation
– Interface graphique
– Entrées/sorties

Mise en œuvre pratique : interface graphique du code en python/gestion des entrées sorties ?

Compilation/debuggage : 2h (Cours+TP).
– Compilateurs
– Options de compilation les plus fréquente
– Directives de compilation
– Outils standards de construction de programme : Makefile, autotools, cmake, scons
– Outils standards de debuggage : gdb, valgrind ...

Application au programme.

Mesures de performance/optimisation : 2h (Cours+TP).
– Profiling
– Instrumentation du code
– Outils de profilage
– Techniques d'optimisation

Application au programme.

Parallélisme : 5h (Cours + TP).
– Introduction aux concepts de la programmation par thread, en particulier d'OpenMP.
– Modèle d'exécution, structure d'OpenMP, partage des données, partage du travail, ordonnancement, synchronisation
– Concepts de base de la programmation par échange de messages - Environnement
– Communications point à point bloquantes ou non bloquantes
– Communications collectives ...

Mise en oeuvre par parallélisation du code.