

# CSE 60531

## **Computational Biophysics**

Spring Semester 2008

*Faculty-in-Charge:* Jesús A. Izaguirre

DeBartolo Hall 246

T H 2:00 – 3:15 PM

This course studies the use of computer modeling and simulation of proteins. Examples that motivate the course include protein folding, docking of ligands to proteins, and conformational changes in general. The emphasis of the course is on the methodology truly useful to the study of biological molecules. Both mathematical and computational aspects of methods are considered. An introduction to the biological issues is also provided. Software especially developed for this course as well as existing tools will be used for the projects, lectures, and tutorials. The course will introduce students to the scripting language Python. There will also be guest lectures by leading experts in the field.

### *Textbook:*

Course packet with notes from instructor and book draft from Robert D. Skeel, Purdue University, *Computational Methods for Biomolecular Simulations*.

### *References:*

Andrew Leach, *Molecular Modelling: Principles and Applications, second edition*, Pearson, 2001.

Tamar Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*, Springer-Verlag, 2002. ISBN 0-387-95404-X.

### *Course Goals:*

At the end of the course, the student should be able to: (1) Setup simulations of proteins and small peptides using existing software tools for the study of problems such as protein folding; (2) Understand the best approaches to do equilibration, sampling, free energy computation, long time dynamics, or kinetics of biological molecules; (3) Understand deterministic approaches to solving some of these problems (e.g., molecular dynamics); (4) Understand stochastic approaches (e.g., Markov Chain Monte Carlo methods); (5) Write scripts in Python to perform analysis of biomolecular simulations; (6) Use databases of molecular dynamics trajectories of simulations (e.g., GEMS) and visualization tools (e.g., VMD) to perform analysis of biomolecular simulations; (8) Understand the mathematical foundations of molecular dynamics and related applications; (9) Undertake some research projects or take more advanced classes in computational biophysics.

*Prerequisites:* Familiarity with a modern programming language is desirable, as well as basic knowledge of probability, calculus, linear algebra, and ordinary differential equations (ODE). Some review tutorials on these topics will be given.

*Topics covered in lecture by week:*

1. Introduction to computational biophysics. Example problems: protein folding, protein-ligand binding, allosteric regulation.
2. Molecular dynamics: Setting up and running molecular dynamics simulations.
3. Computational tasks: steady-state, structure, energetics, kinetics, transition paths.
4. Atomistic models: biomolecular force fields.
5. Coarse-grained models: constraints, implicit solvent, reduced models.
6. Conformational sampling: initialization, equilibration, Hybrid Monte Carlo, thermostating.
7. Conformational sampling: Langevin dynamics, Replica exchange.
8. *Catch up week.*
9. Potentials of mean force: umbrella sampling, WHAM, nonequilibrium sampling.
10. Free energy differences: thermodynamic integration.
11. Kinetics and transition paths: string method, transition path theory.
12. Integrators for NVE and constrained simulation.
13. Integrators for Brownian and Langevin dynamics.
14. Multiscale Integrators: Normal Mode Langevin.
15. Project presentations

*Tutorials (samples):*

- I. Review of math needed for the course (probability, linear algebra, ODE).
- II. Crash course on Python.
- III. Using the Molecular Dynamics Lab (MDL) in Python.
- IV. Using NAMD, ProtoMol, and VMD.
- V. Using distributed systems (clusters and GEMS) to manage distributed simulations and storage.
- VI. Using thermodynamic integration (adaptive biasing force method) in NAMD to compute free energies.
- VII. Using normal modes to explore protein dynamics.

*Homework:* The homework consists of 6 assignments. Some of the assignments require programming. There will also be a final project chosen by the student.

*Laboratory usage:* All students should obtain accounts in high performance clusters at Notre Dame, as well as being able to use the pool for distributed storage in the Department of Computer Science and Engineering at Notre Dame

*Grading:*

Homework assignments	60%
Final exam	20%
Final project	20%