Comprehensive Course Syllabus

Course Title
Computational Science

Course Description:
Computational Science offers an introduction to using computer programming to solve science problems. Students will learn to apply programs they have written to real problems in physics, chemistry, biology, and other sciences. The course will discuss Monte Carlo methodology, minimization, finite element analysis, machine learning, and simulations. Assignments apply object orientation, polymorphism, and data structures to problems such as projectile motion, thermodynamics, reaction rates, natural selection, gravitational interactions, and population dynamics.

I. INSTRUCTOR(S):

- Name(s): Peter Dong

- Office Number(s) (When and where you are available for help.): B117

- Telephone number(s): (630) 907-5476

- Email address(es): pdong@imsa.edu

Meeting Days, Time and Room(s)
in2, A-C days, mods 5-6
in2, B-D days, mods 7-8

Text(s) / Materials:
No textbook will be used. Students will use their own computers, and additional material is available online.

Essential Content:
The relevant Science Team standards are:
A.2 designing and planning investigations and constructing questions which further understanding, forge connections, and deepen meaning. [IL-11.A.5b][NSES-A]
A.3 carrying out investigations that develop skills, concepts, and processes that support and enable complex thought. [IL-11.A.5c][NSES-A]
A.4 using appropriate technologies to collect, analyze and present information. [IL-11.A.5c][NSES-A]
A.6 supporting judgments and constructing models based on evidence. [IL-11.A.5e][NSES-A]
A.8 examining current issues in science and technology. [IL-][NSES-G]

The primary standard is, of course, A.4, since the entire course is predicated on using technology to analyze information. However, this requires students to develop and plan out their investigations (A.2 and A.3). In fact, the planning of the investigation (in this case, the program the students are writing) is more important than the details of the program – the students will learn how to use computer programs to answer scientific questions and make scientific predictions. This leads directly to standard A.6, since they will be constructing computer models, testing them against experiment, and drawing appropriate conclusions.

Some students, based on their prior knowledge, may also gain substantial understanding of a specific content area in chemistry, biology, or physics; for example, those who have not taken EBE will learn a lot about ecology and genetics, and students may get substantial insight into statistical mechanics and other areas such as astronomy or geology.

SSLs and Outcomes:
The primary SSL for this course is, of course, III-A: Use appropriate technologies as extensions of the mind. Supporting this will be I-A: Develop automaticity in skills, concepts, and processes that support and enable complex thought. In this case, I-A refers to programming skills, used to allow students to achieve III-A, which allows them to improve their skills in several other SSLs, including:

I.B Construct questions which further understanding, forge connections, and deepen meaning.
I.D Evaluate the soundness and relevance of information and reasoning.
II.B Recognize, pursue, and explain substantive connections within and among areas of knowledge.
III.C Recreate the beautiful conceptions that give coherence to structures of thought.

SSL I-A will be met by a semester's worth of programming projects, which will help students develop automaticity in basic programming. This will allow them to use computers usefully, as extensions of the mind, to aid scientific research (SSL III-A). Lessons, homework, and projects, will all be focused on the ability to transfer scientific concepts into programming constructs that can be used to answer scientific questions. This is the primary outcome of the class. However, students will also practice coming up with specific questions formulated in a way that computer programs can answer (I-B) when planning projects; they evaluate the accuracy and reliability (or, more often, lack thereof) of computer simulations (I-D) when we test their computer predictions against reality; they use similarity of program structure to explore connections between different branches of science (II-B) when we use the same libraries in different applications; and, overall, they recreate in a computer the conceptions that give coherence to scientific thought (III-C).

Instructional Design and Approach:
This course is designed to be open-ended and project-driven, and is integrative by its very nature. Students are expected to be actively designing and programming for most of each class period. While some assignments will be structured and lead students through the methodology, others will be very open-ended, thus encouraging inquiry. Projects will be oriented around science problems, and thus the class is heavily problem-centered.

Student Expectations:
Students are expected to show maturity and the ability to work well outside of class. Late assignments will be penalized 10% per 24-hour period or fraction thereof. However, students are given five *indulgences* per quarter, by which they can be late without penalty. Tardiness and unexcused absences will be dealt with according to standard IMSA policy. Students are expected to remain on-task on their computer for the entire class period. Students who are consistently and conspicuously off-task are a detriment to the learning environment and may be penalized without warning by loss of points on the relevant assignment.

Students are expected to write their own code themselves. They are encouraged to talk to others, including the teacher, with the understanding that all the actual typing was done individually. Tangential algorithms for a large assignment may come from online sources and should be properly identified in the comments.

Assessment Practices, Procedures, and Processes:
All assessments are programming assignments, categorized as homework or projects. Homework assignments consist of programming tasks of increasing difficulty that build upon one another, labeled as Level I, Level II, Level III, and Challenge. Assignments will be graded with a maximum score based on which level the students successfully complete:

Level I – 70%
Level II – 80%
Level III – 90%
Challenge – 100%

Code will be evaluated on design, clarity, and robustness, in roughly that order. "Design" refers to a sensible translation of physical quantities and variables into computational constructs. This includes lack of cut-and-paste in code, good use of functions, class structure, proper use of inheritance, and well-chosen data structures. "Clarity" refers to proper physical structure of code: good use of space and indentation, good comments, well-named variables, and lack of "magic numbers" or confusing code constructs. "Robustness" refers to program behavior in situations of unusual inputs and includes error checking, array bounds-checking, and exception handling. Note that there is little emphasis on algorithmic structure, which is not the focus of the class.

Homework assignments will generally be weighted equally. Each of the four sub-units will have a final project which will be worth three times a homework assignment. Students will complete the last project during finals period, but there are no timed examinations in this class.

Sequence of Topics and Activities

Unit 1 (7 weeks): Euler's method and finite element analysis with Newtonian mechanics. About three weeks of "boot camp" to go over basics of scientific programming in the framework of simple kinematic behaviors, to get everyone at the same level. Following this, finite element analysis to examine extended objects, culminating with a project on bridge building or marbles.

Unit 2 (5 weeks): Monte Carlo methods with biochemistry. Understanding principles of modeling molecules thermodynamically, then expanding this to chemistry. Use a chemical model to make predictions about a biological system (beta-galactosidase).

Unit 3 (5 weeks): Inference, machine learning, and parallelization with seismology. Use earthquakes to determine the makeup of a planet, and apply a boosted decision tree to separate a signal from a background in Planetary Pirates.

Unit 4 (4 weeks): Evolutionary algorithms with (appropriately) evolution. Create animal behaviors by an evolutionarily trained multi-layer perceptron. Have students' animal behaviors compete against each other in the Hunger Games final project.