Survey of Scientific Computing (SciComp 301)

During this exciting and innovative hands-on course, students will develop strong problem-solving skills in Scientific Computing (SciComp) so they may obtain potential future research internship appointments within world-class organizations. All modern science requires researchers to be adept with writing custom software to solve real-world problems.

No prior programming experience is needed to attend this course. The overall objective is to inform STEM students about the critical need to develop Scientific Computing skills in order to analyze real-world experimental data and to verify theoretical models in biology, chemistry, environmental science, and physics.

SciComp differs in focus from the traditional topics covered in classical Computer Science (CompSci) courses in that SciComp places a greater emphasis on efficient numerical computation, advanced graphical visualization, data storage, simulation, and overall algorithmic efficiency.

For the duration of the course, each student will be given access to their own dedicated remote workstation hosted in the cloud. Students will keep all courseware, programming tools, lab exercises, and exam responses on their remote PC. Students can access their cloud-based SciComp workstation both during class and when off-campus using their personal computer.

Course Goals:

1. Prepare students to conduct interdisciplinary research at world-class institutions
2. Demonstrate how scientific computing impacts all science disciplines
3. Provide patterns for solving real-world science problems by writing custom software
4. Enable students to translate mathematical formulas into correct and efficient code
5. Encourage students to document their research efforts for subsequent peer review
6. Review techniques for the effective visualization of multidimensional data
7. Evaluate optimal methods to store and analyze very large data sets
8. Present scalable approaches to parallel computing on multi-core platforms

Class Schedule:

- A three-hour (180 minute) once weekly class, attended in person for 15 weeks in total
- This will be a 3.0 credit hour course with a total of 45 contact hours with the students
- Additional weekly office hours with the instructor will be available for extra help
- Students will spend an additional three hours per week to complete the lab exercises

Assessments

- **Graded Hands-on Coding Labs** – (65% of grade) During each class students will work individually on several programming exercises which must be completed by the start of the next class. Each lab will be graded according to the correctness of the output and the quality of the source code.

- **Reflective Writing** – (10% of grade) During each class students will maintain free-form digital lab notebook using Zim, a free & open source wiki toolkit for Linux. Within their Zim file, students will express their own understanding of the material for each session, and also annotate their research approaches for solving the lab exercises. The instructor will review the quality and effort invested by each student to develop meaningful lab notebooks and will provide feedback on how to improve their collaborative idea sharing with their peers.

- **Written Tests** – (15% of grade) Throughout the course students will complete three (3) written take-home exams. These exams will be a combination of completing in-depth analysis of prior hands-on labs for the review period, as well as writing new code to solve related scientific problems. Students are free to use any available online resource or code repository, provided an accurate and detailed attribution is made as to the source website & author of the original code. It is expected that the preponderance of the student's answers will represent their own work. Undocumented code plagiarism from external sources will result in a failing score.

- **Student Final Presentations** – (10% of grade) Over the duration of the course students will be paired to work together on a final coding project of their own design and interest. The topic and approach for the final project must be approved by the instructor prior to the team starting the actual programming. At the final class, teams will be given five minutes to present their working solution using a standard discussion outline provided by the instructor. This exercise simulates how scientists collaborate in colloquia to share the results of their research efforts. It also helps build their public speaking skills and boosts their self-confidence.

Prerequisites:

1. Successful prior completion of course in trigonometry

2. No prior programming experience is required. All of the necessary software development skills will be taught as part of the course. The programming language taught will be C++/14 using Code::Blocks running on Ubuntu Linux machines hosted in Amazon's cloud.

Course Syllabus: