Course Description:

This course is designed for graduate students who are developing the methods, and using the tools, of scientific computing in their research. With the increased power and availability of computers to perform massive and complex simulations, computational science and engineering as a whole has become an integral part of research that complements experiment and theory. This course will teach students the necessary “skills” to be effective computational scientists and how to produce work that adheres to the scientific method. A broad range of topics will be covered including: software engineering best practices, computer architectures, computational performance, common algorithms in engineering, solvers, software libraries for scientific computing, verification and validation, and how to use all the various tools to accomplish these things.

Course Objective

Upon successful completion of the course students shall be able to

- develop and run software in Linux,
- write code in multiple languages,
- use compilers and Makefiles,
- write their own linear solver
- compile and use third party libraries,
- work in software projects with other individuals,
- develop version controlled software,
- implement automated testing in a software project,
- increase the computational performance of their software
- write code that uses MPI and/or OpenMP parallelism,
- perform simulations on high-performance computing resources,
- debug programs more efficiently

Course Schedule

Date	Lecture	Lab	Topic
	1		Course Overview & Introduction to Linux
	2		Programming Languages: C, C++, Fortran
		1	*Introduction to Linux*
	3		Scripting with Bash and Python
		2	*Scripting*
	4		Elements of Development: Configuring, Compiling, Linking
	5		Tools of the Trade: Version Control, Dev. Env
		3	*Introduction to Great Lakes and Git*
	6		Algorithms for Linear Algebra
	7		Solving Linear Systems (Part 1) Classical Methods
		4	*Matrix-Matrix Multiply and Third-Party Libraries*
	8		Solving Linear Systems (Part 2) Krylov Methods
	9		Solving Ordinary Differential Equations
		5	*PETSc and Krylov Methods*
	10		Software Engineering
	11		Object Oriented Programming and Design
		6	*Workflows in Practice*
	12		Architecture and Design
	13		High-Level Design and C++
		7	*More Workflow and Sprint Planning*
			FALL BREAK
	14		Serial Architecture, Performance, and Optimization
		8	*Micro-Benchmarks and Measuring Performance*
	15		Parallel Architecture and Performance
	16		OpenMP
		9	*Parallel Computing: OpenMP*
	17		The Message Passing Interface
	18		Advanced MPI
		10	*Parallel Computing: MPI*
	19		Heterogeneous Architectures
	20		Programming models for GPUs
		11	*Hardware Abstraction with Kokkos*
	21		Testing, Verification, and Validation
	22		How to write a Unit Test
		12	*Automated Testing Infrastructure*
	23		Using Jupyter Notebooks with HPC
	24		Profiling and Debugging Tools
	25		Retrospective and Miscellaneous
	OPEN LAB - Work on Term Projects		
	26		Term Project Presentations
	27		Term Project Presentations

Optional Course Modules

Topic			Description
Testing, Testing, Testing	Lecture 1		Testing, Verification, and Validation
	Lecture 2		How to write a Unit Test
		Lab	*Automated Testing Infrastructure*
Data and Mesh Libraries	Lecture 1		Data Format Libraries: HDF5, NetCDF, SILO
	Lecture 2		Mesh Libraries: Libmesh, Exodus, others
		Lab	*Working with Data Libraries*
Package Management & Containers	Lecture 1		Package and Dependency Management with Spack
	Lecture 2		Containers: Docker and Apptainer
		Lab	*Spack and Apptainer on Great Lakes*
Python for HPC	Lecture 1		Using Jupyter Notebooks with HPC
	Lecture 2		Packages for Scientific Computing
		Lab	*TBD*
Debugging and Profiling Tools	Lecture 1		Debugging: DDT, GDB, and Valgrind
	Lecture 2		Performance: MAP, HPCToolKit, TAU
		Lab	*Make it work; Make it fast -- Debug and Optimize*

Anticipated Homework Assignments

HW	Description	Supporting Lectures	Due Date
1	LaTeX and Programming in C/C++ and Fortran	1,2	
2	Some Linear Algebra Kernels	2,4,6	
3	Linear Algebra Solvers	2,4,6,7,8	
4	Workflows	8,10,11	
5	Extra-Credit		

Anticipated Lab Assignments

Lab	Description	Supporting Lectures	Due Date
1	Hands on walkthrough of Linux.	1	
2	Bash and Python Scripting	2,3	
3	Hands on walkthrough of Great Lakes	1-4	
4	Matrix-Matrix Multiply and TPLs	4-6	
5	PETSc and Krylov Methods	4,7-9	
6	Workflows	5,6,10,11	
7	More Workflows	10-13	
8	Micro-benchmarks and performance	14	
9	OpenMP simulated annealing parallel programming.	2,7,15,16	
10	MPI simulated annealing parallel programming.	2,4,7,15,17	

Project Assignments

Deliverable	Description	Due Date
Proposal	3-5 page document	
Presentation	10-20 minute presentation	
Report	10-20 page typed document	

Associated Readings and General References:

Lecture	Topic
1	None
2	Modern Fortran C/C++ programmer's reference
3	Bash Quick Start Guide Learning Python
4	None
5	Pro Git
6-9	Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide Finite Difference Methods Finite Element Methods The Method of Weighted Residuals and Variational Principles Multigrid Methods Matrix Algorithms Vol. 1 and Vol. 2 Iterative Methods for Sparse Linear Systems Model Reduction and Approximation Accuracy and Reliability of Scientific Computing Accuracy and Stability of Numerical Algorithms LAPACK User's Guide ScaLAPACK User's Guide Numerical Recipes in Fortran (Online PDF) Numerical Recipes in C (Online PDF)
10-11	Scientific Software Design: The Object-Oriented Way The Unified Modeling Language Design Patterns Code Complete Agile Development in the Real World
12-13	Numerical Linear Algebra for High-Performance Computers Performance Optimization of Numerically Intensive Codes
14-16	Patterns for Parallel Programming Parallel Processing for Scientific Computing
	Using OpenMP
	Using MPI
	Using Advanced MPI
	Verification and Validation in Scientific Computing
	Better Scientific Software