MATH 676

Finite element methods in scientific computing

Lecturer:	Prof. Wolfgang Bangerth Blocker Bldg., Room 507D (979) 845 6393 bangerth@math.tamu.edu
Lecture:	Tuesdays + Thursdays, 9:35–10:50am Blocker 122
Office hours:	Wednesdays 10:00–12:00am and by appointment

Course Outline

The course is intended to give students a perspective on the practical aspects of the finite element method, in particular on how typical finite element software is structured, on algorithmic details of how to efficiently implement it, and the pre- and postprocessing steps necessary in the scientific computing workflow.
One of the most important lessons learned over the past decade is that due to their size, modern numerical software can't be written from scratch for each new project, and must instead build on existing software libraries that handle most of what constitutes the finite element method as well as linear algebra and parallel communication. Applications then only have to implement things like bilinear forms, outer nonlinear solution loops, and linear solvers specialized to the application.
The course will use a large Open Source finite element library and other open source tools for the entire workflow. The library, deal.II, can be found at `http://www.dealii.org` and is developed mainly at Texas A&M University in collaboration with other researchers worldwide. It is used as the standard research and teaching tool for numerical computing in the Department of Mathematics. It supports the discretization of arbitrary partial differential equations and runs on machines from laptops to supercomputers with more than 10,000 processors.
The first part of the course will be used to review the basic mathematical concepts used in this software, such as finite element theory and iterative solvers. The rest of the course will be used for projects in which students are guided through the implementation of an application related to their research project or interests. Part of this will be teaching the use of modern software engineering practices, such as the use of revision control management systems (e.g. Subversion), writing documentation, and writing tests for automated testsuites. The goal is the development of a code that a) helps in the research of a student, and b) may be used as the starting point for future generations of students. It may also be converted into a tutorial for deal.II.

An outline of topics to be covered is as follows:

- Basics of finite element methods
- Structure of finite element codes
- Assembling linear systems of equations and algorithmic aspects
- Iterative solvers for large linear systems
- Adaptive mesh refinement
- Vector-valued and mixed problems
- Nonlinear problems
- Postprocessing, visualization, and parallelization aspects
- Collaborative software development, e.g. through the use of subversion
- Software engineering practices for large-scale software, e.g. automated build and test systems, tools for documentation
- Additional topics may be chosen based on student interests.

This course is intended for students of mathematics involved in research in numerical methods as well as students in the engineering disciplines.

Prerequisites

- Knowledge of the finite element method. This is covered by MATH 610, though we will probably need little of the theory developed there. An engineering equivalent of this course would definitely be sufficient. Good students of MATH 609 should also be able to read up quickly on how the finite element method works, though they may lack an understanding of why.
- Basics of iterative solvers. MATH 609 covers some of this material, and MATH 639 covers it in detail. Good students of MATH 609 should do fine, though.
- Good knowledge of C++. Students without adequate C++ programming skills will not be able to benefit from this course, and in contrast to content material, the experience needed to write programs can't be taught in a few classes at the beginning of the semester. All successful large-scale software packages are written in advanced programming languages, and students should have experience in object-oriented programming and preferrably the use of templates in C++.

Course Schedule
Here is a rough outline and schedule of the course. Actual allocation of weeks may differ and will depend on student interest and the projects given to students:

Week 1, 1/17	Getting and installing deal.II. Basics of finite element programming.
Week 2, 1/24	Review of C++ templates. The step-1 and step-2 tutorial programs. Project presentations.
Week 3, 1/31	Project presentations. Step-3 and step-4.
Week 4, 2/7	Step-5, step-6.
Week 5, 2/14	Visualization, collaborative software development, defensive programming, step-7, basics of vector-valued problems.
Week 6, 2/21	Vector-valued problems: setting up block matrices and vectors, partitioning degrees of freedom, deriving block solvers by considering matrices only as linear operators; using templates to describe ¡i¿concepts¡/i¿ instead of actions, step-20, step-22.
Week 7, 2/28	Step-20, step-22, time-dependent problems: classification and examples.
Week 8, 3/7	Time-dependent problems: the heat equation, explicit and implicit schemes; the wave equation: explicit and implicit schemes.
Week 9, 3/14	Project work.
Week 10, 3/21	Mid-semester presentations.
Week 11, 3/28	The ParameterHandler class to deal with run-time parameters, multithreading.
Week 12, 4/4	Differential-algebraic equations, IMPES schemes; time step choice in transport equations.
Week 13, 4/11	Parallelization, project work.
Week 14, 4/18	Project work.
Week 15, 4/25	Project work.
Week 16, 5/2	Project work; final project presentations on Friday May 6th, 2011, 12:30-2:30pm.

Literature

To the best of my knowledge, there isn't much in terms of books on writing large-scale software for finite element simulations. What is there often treats a single software package; for example, there are books about FEMLAB and DiffPack, two commercial finite element packages. For the packages to be used in class, there are no such books, but deal.II has several thousand pages of documentation. It can be found at the deal.II homepage at `http://www.dealii.org/`.
For the other topics, there are good books. There is a plethora of books about the theory of finite elements; for iterative solvers of linear and nonlinear systems, the following books are commonly referenced:

- Y. Saad, "Iterative Methods for Sparse Linear Systems" (an exhaustive book on the theory of iterative solvers)

- R. Barrett et al., "Templates for the Solution of Linear Systems" (a short book explaining the implementation of a variety of linear solvers)
- C. T. Kelley, "Iterative Methods for Linear and Nonlinear Equations"

The use of Subversion is explained in

- C. M. Pilato et al., "Version Control with Subversion" (this book is available online at `http://svnbook.red-bean.com/nightly/en/index.html`)

Other software design techniques such as unit testing and automated build systems are treated in many Computer Science texts, but what is needed will also be covered in class.

Webpage
Homework assignments and other course information will be posted at the course webpage

`http://www.math.tamu.edu/~bangerth/teaching.html`

Exams + Grading
Final course grades will be determined from the project every student gets, although projects can also be worked on in small groups. Projects will consist of writing a finite element solver for a particular equation that will generally be chosen from the area of interest of each student. The determination of the grade will be based on the following criteria:

- Sophistication of the application beyond the program from which it was started
- Extent of documentation in the code
- Extent of the documentation surrounding the program, i.e. description of the equation and its properties, description of the principles used in the implementation, and documentation of worked-out examples computed with the program
- Quality of the presentation at the end of the semester.

As an example for projects, take a look at the deal.II tutorial programs at `http://www.dealii.org/developer/doxygen/tutorial/index.html`. If students are interested, good enough projects will be published as part of the library and distributed with future versions (see for example the step-21, step-24, and step-25 tutorial programs that were created by students of a prior class). Of course, the students will then be credited for their work.
Students will be expected to report on the progress of their project at one point during the semester, and give a final presentation at the end.

Incompletes: I will consider giving an incomplete if you have successfully completed all but a small portion of the work of the course, and are prevented from

completing the course by a severe, unexpected event. Simply being behind work is not a reason for an Incomplete, though; in that case you should consider dropping the course.

S/U grades: If you are registered S/U your grade will be 'S' if your letter grade is C or above, and 'U' otherwise.

Policies

Academic integrity: The usual rules of academic intregrity apply. In particular, the Aggie Honor Code "An Aggie does not lie, cheat or steal, or tolerate those who do" should be selfevident, see

`http://www.tamu.edu/aggiehonor.html`

Students may, and are encouraged to, work together and discuss homework problems with each other. However, copying work done by others is an act of scholastic dishonesty and will be persecuted to the full extent allowed by University policy.

Disabilities: If you have a disability and need special assistance, please contact me so we can make accomodations. The Americans with Disabilities Act (ADA) is a federal anti-discrimination statute that provides comprehensive civil rights protection for persons with disabilities. Among other things, this legislation requires that all students with disabilities be guaranteed a learning environment that provides for reasonable accomodation of their disabilities. If you believe you have a disability requiring an accomodation, please also contact Services for Students with Disabilities, Koldus 126, 845-1637.

For other policies and other information, please read

`http://www.math.tamu.edu/teaching/operationspg.html`