# MATH 676

# Finite element methods in scientific computing

| | |
|---|---|
| Instructor: | Prof. Wolfgang Bangerth<br>Blocker Bldg., Room 507D<br>(979) 845 6393<br>`bangerth@math.tamu.edu` |
| Lecture: | Tuesdays + Thursdays, 11:10–12:25am<br>Blocker 123 |
| Office hours: | Thursdays 9:30–11:00am 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:

- Installation and setup of deal.II
- Basics of finite element methods
- Structure of finite element codes
- Eclipse – an Integrated Development Environment
- Assembling linear systems of equations and algorithmic aspects
- Iterative solvers for large linear systems
- Visit – a modern visualization tool
- A bit of background on C++

- Adaptive mesh refinement
- Vector-valued and mixed problems
- Software design and data structures
- Documentation and the doxygen tool
- Nonlinear problems
- Time dependent problems
- Hyperbolic problems
- Collaborative software development and the subversion tool
- Postprocessing, visualization, and parallelization aspects
- An overview of linear solvers
- An overview of preconditioners
- An overview of the workflow in scientific computing
- Software engineering practices for large-scale software

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.

## Format

This course consists of three components: lectures, project work and a journal.

- *Lectures:* Past experience has shown that the most important point of this class is that I have time to work with you on your projects. Consequently, the *lectures* have been recorded ahead of the class schedule by KAMU. All students are *required* to watch these lectures outside of class. I will not answer questions that have been answered in the video lectures. Students will take turns summarizing these video lectures at the beginning of each class period and we will discuss open questions.
- *Projects:* Each student will be assigned a project related to his or her research. You will work on your project during class, when I will be available to help you, as well as outside class. Each student will give a short presentation at the beginning of the semester outlining her or his research and upon which I will base your project assignment. There will also be a short progress report presentation about midterm and a presentation at the end of the semester outlining the results you have achieved.
- *Journal:* The journal is the computational scientist's equivalent to a lab book and serves the purpose of recording your progress, notes, and questions to give you the opportunity to reflect on and monitor your learning process. In it you will also keep a record of when you watched video lectures. I expect that you will use the journal as a place to reflect on what and how you have learned throughout the semester regarding the course content and your project. Taking the time to do this will often allow you to see patterns and develop insights that might otherwise go unnoticed.

  Your journal should contain the following:

  - A table of contents that includes entry titles and dates. Keep this at the front of your file and update it periodically. (Google Drive, which we will use for this, allows you to add a table of entries automatically linked to section headings of the document.)

  - The date of each entry. Include the time if you make multiple entries in a single day.
  - A record of the lecture videos you watched. For these entries, include a summary of the three most important points from this lecture and two or three questions you still have. These entries will serve as the foundation for our class discussions.
  - Project log and reflective entries. Record when you work on your project, what you did during that time, how it worked (or didn't work), notes for what you want to try next etc. You should also spend some time reflecting on how the work is going, your progress, strategies, etc. If you were describing your project/work to a friend, what would you say? These entries will be particularly useful for your final paper and reflection essays.

Twice during the semester (at the same time as your midterm and final presentations) you will submit an essay, about 1-2 pages, that summarizes an important insight you have had regarding this class and how you arrived at that insight. Examples might include understanding a new technique to debug programs; discovering another reason why version control systems are useful; etc. To inform these essays, read back over your journal and look for patterns, aha moments, or anything that stands out to you as particularly important. I encourage you to reference specific journal entries as part of your essay. I will periodically review your journals and leave notes in them. I expect to see an entry for every video lecture plus 2-3 independent entries per week. Please keep in mind that my access to your journal means that you shouldn't include anything you don't want me to read.

To facilitate the journals, please open a document on Google Drive, name it *MATH-676-lastname-firstname*, and share it with `bangerth@gmail.com`.

## 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 preferably the use of templates in C++.

## 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 standard reference for the C++ language that we will use extensively in this class is

- B. Stroustrup, “The C++ Programming Language”.

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.co`

There are a number of books on the use of the Eclipse Integrated Development Environment. 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.
You are not required to buy or use any of these books. The list above is simply as reference if you want to look up things or learn them in more detail.

## Webpage

Homework assignments and other course information will be posted at the course webpage
`http://www.math.tamu.edu/~bangerth/teaching.html`

## Grading

Final course grades will be determined based on the project every student gets, although projects can also be worked on in small groups with instructor approval. 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.
The journal will be graded by completeness as well as the depth of reflection in the two essays. The journal entries should demonstrate consistent and sustained commitment to the course.
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.
The overall grade will be determined by the project (80%), the presentations (10%) and the journal (10%).

*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

*Late work:* I will not accept late work unless we have a prior agreement. If you need an extension to finish your work, talk to me!
*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`