Math 615 Numerical Analysis of Differential Equations

Spring 2014 UAF

Instructor: **Ed Bueler**
Phone: 474-7693
eMail: elbueler@alaska.edu
Office: Chapman 301C, hours at bueler.github.io/OffHrs.htm

Class Time: MWF 1:00--2:00
Classroom: Reichardt 204
Web Site: bueler.github.io

Course Description: 3.0 credits. We cover methods for numerical approximations to partial differential equations (PDEs) and related problems on computers. There will be some coverage of numerical methods for ordinary differential equations (ODEs). PDEs are the underlying structure for most problems of flow, fields, thermodynamics, deformation, quantum mechanics, chemistry, and so on, but also finance and population dynamics. We do mathematical analysis of these numerical methods, with both practical and abstract approaches to this numerical analysis: (i) how are these methods implemented, verified, and used? (ii) when are they stable? and (iii) do we know in advance that they converge? The emphasis will be on finite difference methods for PDEs. I will only gloss other methods (e.g. finite volume and element) for PDEs.

Lectures will include *Matlab* demonstrations whenever I can fit them in. I will help you get started with *Matlab*, but you must show initiative in learning actual numerical computation. Homework assignments and a student-chosen project will include actual implementations in *Matlab*. Abstract and precise thought is, however, essential to make choices among numerical methods for solving major problems. Thus all homework assignments will have mathematical exercises, and in these you will be asked to "show" and "prove". Formal proof style is not important, but you'll need to give clear presentations of sufficiently-general logical arguments.

We will think in terms of vectors and matrices. We will not be satisfied with seeing lots of numbers or pretty pictures from our computer programs. Instead we will be concerned with the degree to which our numbers represent what we claim they approximate. We will seek the underlying linear algebra structure of PDE problems. The course will include some nonlinear examples, for which one uses a sequence of approximating linear problems.

Goals: At the end this course you will not be a professional numerical analyst. But you will be able to evaluate and use many numerical tools for solving scientific and engineering problems, and you will be able to code some of the basic methods (i.e. for the purpose of prototyping more serious solutions). Furthermore you will have the mathematical start needed to take the next steps to learn the finite element method, spectral methods, matrix methods, and broader scientific computing.

Calendar and Course Webpage: A day-to-day (tentative) schedule for the semester is at the course webpage bueler.github.io/M615S14/.

Prerequisites: *Informally:* undergraduate ordinary differential equations, undergraduate linear algebra, exposure to the basic ideas of numerical analysis, and exposure to Fourier series and separation of variables (for solving the classical linear PDE boundary value problems). Also some exposure to computer programming. *Formally*: "Prerequisites: CS F201, MATH F310, MATH F314, MATH F421, MATH F422 or permission of instructor."

Textbook: The **required** text is K. Morton and D. Mayers, *Numerical Solution of Partial Differential Equations*, 2nd ed. Cambridge Univ. Press 2005. We will cover chapters 1, 2, 3, 4, 5, and 6.

Your Grade = Homework + Project + Midterm: **Sixty percent** of the course, and the grade, will be based on **weekly homework assignments**, including a final assignment which will be worth double an ordinary assignment. You will be asked to think abstractly on some problems and to use *Matlab* on others. (Expect to write programs about a half-page long.)

It is assumed that students in this class have in mind, or can acquire, specific modelling problems in applied fields which can be used for a **project**. These will mostly, but not exclusively, be PDE problems. Students often use a simplification of a thesis/dissertation project, for instance. I am eager to help and advise on choosing and refining such problems. **Twenty-five percent** of the grade in the course will be on the project. Two project assignments will be given, the first part a preparatory stage due midsemester, and the remainder due at finals time. Both mathematical analysis and actual numerical computation are required on your project.

Finally, there will be a one-hour in-class **midterm exam** worth only **fifteen percent** of the course grade. The purpose is to give a midsemester opportunity to review basics before expanding our goals in the second half.

The **course grade** will be determined from

homework+project+midterm

according to the schedule at right --->

I will use plus/minus grades as indicated.

Percent	Grade
91 --100 %	A
88 -- 90 %	A-
84 -- 87 %	B+
76 -- 83 %	B
73 -- 75 %	B-
69 -- 72 %	C+
57 -- 68 %	C
41 -- 56 %	D
0 -- 40 %	F

Policies and makeup exams: The department has specific policies on incompletes, late withdrawals, and early final examinations, etc; see http://www.uaf.edu/dms/policies/ You are covered by the UAF Honor Code. I will work with the Office of Disabilities Services (208 Whitaker, 474-5655) to provide reasonable accommodation to student with disabilities.

Programming in the course: You will use *Matlab*, *Octave*, or *pylab* (= *python+scipy+matplotlib*) in this course. These will be used in homework problems and in projects. *Matlab* is commercial while *Octave* and *Pylab* are free and open source. *Octave* is a clone of *Matlab* so the same programs will run in both. Programs in *Matlab/Octave* will appear on my website (and occasionally in *pylab*).

Copious online resources are available for learning *Matlab/Octave/pylab*. They are all languages designed to do numerical analysis coursework. Mathematical and graphical inputs and outputs are easily manipulated. Matrices appearing in problems can be easily analyzed. Many of the operations appearing in numerical problems are natural and quick and require less work than in compiled languages like *C* or *FORTRAN*.