ENME745 / ENME489J
Numerical Methods in Engineering
Fall 2021

Last updated: July 15, 2021

Instructor

Johan Larsson
jola@umd.edu; office: EGR3149; office hours: Thu 1245-1400.

Course objectives

This course covers the fundamental aspects of how to apply analytical mathematical concepts to discrete data. The course is aimed at graduate students and senior undergraduate students in any area of engineering, and treats the material in a general manner that is not specific to any application or field of specialization.

The material covered is essential and foundational for students that will engage in computational research, but is also meant to be highly useful to students engaging in experimental work.

Learning outcomes

Learn standard root-finding methods. Learn the basic principles and methods for computing integrals and derivatives from discretely known data. Learn basic curve-fitting methods. Improve the ability to write computer code. Learn how to solve initial-value and boundary-value ordinary differential equations. Gain a basic understanding for how to solve simple partial differential equations. Gain an appreciation for the importance of verification in scientific programming.

Prerequisites

Students should a working knowledge of calculus (partial derivatives, integration, ODEs), complex variables (e.g., $z = \exp(i\theta) = z_r + iz_i$), and linear algebra (vectors, matrices, eigenvalues and eigenvectors, diagonalization of matrices), commensurate with an entry-level graduate course in engineering mathematics.

Students should also have a working knowledge of some programming language, e.g., Matlab, Python, or similar. Students need *not* be expert programmers – but they do need a willingness to take on challenging programming tasks (with help from the instructor and peers in the class, of course).

Course format

The material will be covered during two lectures of 75 mins each per week.

There will be weekly homework, consisting of a written portion (deriving equations, analyzing numerical methods, etc) and a coding portion where students must solve problems using Matlab, Python, C/C++ or any other suitable language of their choice.

The nature of the material covered is such that it can only be learned fully by actually implementing the methods and seeing different outcomes; thus there will be a greater emphasis on homework than is common. Key concepts that homework is meant to bring to life will be covered and discussed in lectures on an ongoing basis.

Undergraduate vs graduate students

This course has students with very varied backgrounds. Some are undergraduate students, others are graduate students that specialize in experiments, while others are graduate students that specialize in computational work. This makes it challenging to design a course that is meaningful and stimulating for all students.

The lectures will mostly cover the basic material, in a way targeted more towards the students with less familiarity of numerical methods. The homework will include both more basic questions (that should be solved by everyone) and a few more challenging questions that are meant mainly for the more advanced students. In addition, there will be a non-mandatory weekly discussion session aimed at the more advanced students. This group may be asked to implement additional methods, read additional papers or materials, or similar, in advance of the discussion sessions. The idea is to give the more advanced students an opportunity to progress further.

The discussion session will be on Thu 1400-1500. It is completely optional and absolutely not mandatory.

Evaluation

The homework will be graded on the basis of completion, not correctness. Two take-home exams will be given and graded on the basis of correctness. The final grade is a combination of the homework credit and the exam scores.

In both homework and exams, some questions will be required only by the graduate students.

Academic integrity and collaboration

Collaboration is encouraged in this class. Having said that, every student must complete their own homework by themselves (e.g., write their own computer code, produce their own plots, derive their own equations, etc). In other words, students are allowed (and encouraged!) to work together when figuring things out, and to ask each other questions etc, but must make sure that they do the homework themselves without copying.

Internet resources that explain concepts (like Wikipedia and some Youtube videos) can be excellent learning aides – they explain the same thing in different words which may help students' understanding. These types of internet resources are absolutely allowed. However, internet resources that provide solutions to problems are not allowed and are considered cheating.

The take-home exams must be completed without any collaboration.

Textbooks

Recommended texts:

- "Numerical Methods for Engineers", S. C. Chapra and R. P. Canale, McGraw-Hill (easy-to-understand introduction to the subject).
- "Fundamentals of Engineering Numerical Analysis", P. Moin, Cambridge University Press (covers much of the material in the course in a concise way, but starts at a higher level – suitable for advanced graduate students with some background in numerical methods).
- "Assessing the Reliability of Complex Models", National Research Council (available for download from the NRC website under "Resources").

Texts for further/deeper reading, suitable for students who are particularly interested in the specific areas:

- "Numerical Linear Algebra", L. N. Trefethen and D. Bau, SIAM Press (pedagogical coverage of numerical algorithms for solving linear algebra problems).

- "Data Analysis: A Bayesian Tutorial", D. S. Sivia, Oxford Science Publications (nice book for those interested in Bayesian inference, sufficiently well written to be read purely for pleasure).

Topics and tentative schedule

Approximate number of 75-minute lectures spent on each topic is given in square brackets.

1. **The basic building-blocks of numerical methods** [12]
   (a) **Mathematical foundation:** Taylor expansions; errors and convergence.
   (b) **Root-finding:** bisection; Newton-Raphson; the secant method.
   (c) **Interpolation:** polynomial interpolation and extrapolation; cubic splines; least-squares fits; multi-dimensional interpolation.
   (d) **Differentiation:** finite differencing; Taylor series analysis; non-uniform grid; the modified wavenumber; Padé approximation.
   (e) **Integration:** trapezoidal rule; error analysis; Richardson extrapolation; Gauss quadrature.
   (f) **Expansion in basis functions:** discrete Fourier transform; other basis functions.
   (g) **Filtering:** wavenumber analysis and transfer function.
2. **Ordinary differential equations (ODEs)** [6]
   (a) **Initial-value problems, scalar:** Euler's forward and backward methods; Runge-Kutta methods; linearization; error and stability analysis.
   (b) **Initial-value problems, systems:** linearization; analysis through eigendecomposition; spectral radius; stiffness.
   (c) **Boundary-value problems:** the shooting method; matrix formulation; iterative solution techniques.
3. **Partial differential equations (PDEs)** [8]
   (a) **The finite-difference method:** semi-discretization; stability and accuracy analysis; conservation properties; boundary conditions and stability; boundary conditions and reflections; spectral radius.
   (b) **Applications:** hyperbolic, parabolic and elliptic scalar problems; hyperbolic systems.
4. **Verification and validation (V&V)** [3]
   (a) **Error sources:** influence of grid-spacing, domain size, artificial boundary conditions; modeling errors.
   (b) **Code verification:** analytical solutions; linearized eigenmodes; the method of manufactured solutions.