SYLLABUS

NUMERICAL ANALYSIS I

FALL SEMESTER 2017

http://www.phys.uconn.edu/~rozman/Courses/m3510_17f/

Last modified: September 25, 2017

Course description: The overall goal of numerical analysis is the design and analysis of techniques to give approximate but accurate solutions to hard problems. MATH 3510 *Numerical Analysis I* teaches this analysis associated with direct methods of solution of linear systems, zeros of non-linear equations, approximation and interpolation, numerical integration, initial value problem for ordinary differential equations, with attention to generation and propagation of numerical errors and to computational speed.

Lectures: TuTh 5:00 – 6:15, Laurel Hall 202

Course webpage: http://www.phys.uconn.edu/~rozman/Courses/m3510_17f/

Instructor: Michael Rozman

email:	rozman@phys.uconn.edu
phone:	860 486 5827
office hours:	Mo 1:00 PM – 2:00 PM in MONT130,
	Tu 6:15 PM – 7:15 PM in LH202/LH305,
	We 5:00 PM – 6:00 PM in MONT130,
	by appointments

Textbook:

R. L. Burden and J. D. Faires, *Numerical Analysis*, 10 ed., Cengage Learning, 2015

Exams: Two closed book midterm exams and a closed book *cumulative* final exam

Grading scheme: The course grade will be calculated using the following scheme.

Homework projects	35%
2 Midterms	35%
Final exam	30%

Class schedule: this is a *preliminary* schedule.

Week(s)	**Subject**
1-2	Matlab ~~and Julia~~ programming
3-5	Direct methods for solving of linear systems
	Midterm I - Tue, Oct 3
6-7	Solution of nonlinear equations
8-10	Interpolation and approximation
	Midterm II - Thu, Nov 2
11-12	Numerical integration
13	**Thanksgiving recess**
14-15	Initial value problem for ODE

Communications: talking in person is the preferred method to contact the instructor; email is the next best option.

- please include the tag ``[math3510]'' (without quotes, no spaces) in the subject of your email, e.g. "[math3510] midterm II review session"
- please no emails with attachments unless requested by the instructor. Use *UConn File DropBox* `https://dropbox.uconn.edu/dropbox` or *UConn FileLocker* `http://web2.uconn.edu/filelocker/` for submitting large files

Homework: Homework assignments submitted before or on the due date may be returned (at the discretion of the instructor) for corrections after initial grading.

Homework assignments are not accepted after the solutions had been discussed in class and/or had been posted online. Individual emergencies can be accomodated by extra credit assignments.

You are welcome to discuss the homework's problems with others in order to better understand them but the work you turn in must be your own. In particular, you must run your own calculations (where applicable) and communicate and explain the results in your own words.

Assignments that are hard to understand are also hard to grade correctly, therefore: (a) use words and pictures to supplement your equations; (b) work must progress linearly down the page – recopy solutions that are too nonlinear.

Requirements for written assignments:

- Use letter-size paper. Use only one side of each sheet.
- $\boxed{\text{Box}}$ your final answer(s).
- Staple your notes together, (i.e. no paper clips, torn or folded corners) with the assignment cover page (if applicable).

Highly recommended: make copies of homework assignments for your own files.

Computer programming:

Homework projects require writing simple computer programs. You may use any programming language you like (however, since the instructor needs to read and run your code, clarify in advance the case of "exotic" languages). If you are not sure which language to use, here are some thoughts to help guide your decision[1].

The C programming language and its close relatives and offsprings (C++, Java, C#, etc.) are the widely used programming languages. Programming in these languages offers you maximal flexibility in terms of using your computer's resources. Programs written in C (which is a *statically typed compiled language*) typically run significantly faster than programs written in matlab, python, or other *dynamically typed interpreted* languages. Moreover, this option offers you the widest range of interoperability with existing codes. If you are planning to do a lot of programming or to work with existing codes in your future, you are strongly recommend to acquire a working experience with C/C++. However, if you have not done much or any C/C++ programming yet, you may experience a

[1]Shamelessly appropriated from `http://homerreid.dyndns.org/teaching/18.330/`.

rather steep learning curve, and you may not wish to make that time investment for MATH 3510.

Matlab is a common choice for academic numerical programming. It is much easier to get started programming in Matlab than in C/C++. Your programs will run more slowly but this factor is not significant for the programs you need to write in MATH 3510. If you are planing to take MATH 3511 Numerical Analysis II, you will need Matlab programming skills for that class.

The problem with Matlab is that it is a commercial product that is too expensive for anybody outside an academic or corporate environment. This means that (a) you may find yourself without access to the tool once you are out of college, and (b) codes you may write and want to share with others may be of limited utility, because most people in the world don't have Matlab.

Julia combines many of the best features of the previous two options. Julia is a language developed over the past several years, mostly at MIT. It is as easy to use as Matlab, but it has the critical advantage of being free, open-source software, so there is no fear of ever losing access to it, nor of not being able to share your codes with someone who doesn't have it yet. It is significantly faster than interpreted languages (which again, this is not a crucial concern for MATH 3510) and has an enthusiastic and growing base of users and contributors.

The main problem with Julia is that it is relatively new and hence a little rough around the edges. Using Julia has a bit of a startup/wild-wild-west feel to it which may not appeal to all students.

Recommended reading:

- C. Moler, *Numerical Computing with MATLAB*, MathWorks, 2008 (online, last accessed on August 2, 2017)