Math 4610 Fundamentals of Computational Mathematics Fall Semester 2017

Instructor: Joe Koebbe
Office: ANSC 209
Office Hours: 9:30-10:20 TTh and 12:00-1:00 TTh or by appointment.
email: Joe.Koebbe@usu.edu
webpage: http://www.math.usu.edu/ koebbe
Office Phone: 435-797-2825

Important Contact Information: If you call my office phone number, leave a message on voice mail. Voice mail messages are automatically forwarded to my email and I can listen and respond at home. The best way to contact me is directly through email. I read email multiple times each day.

USU Course Catalog Description for this Course:

MATH 4610 - Fundamentals of Computational Mathematics 3 credits Course presents fundamental topics common in computational mathematics and problem solving, including: truncation and round-off error, algorithms to find solutions of nonlinear equations, numerical solution of linear systems by direct and iterative approaches, interpolation methods, numerical differentiation and quadrature rules. Prerequisite/Restriction: MATH 1220, MATH 2250, or MATH 2270 with a C- or better. Ability to program with a high-level computer language (C/C++, Python, Fortran) Semester(s) Traditionally Offered: F

Textbook: "A First Course in Numerical in Numerical Methods. " by Uri Ascher and Chen Greif

General Comments/Polocies on the Course: This course covers the development and implementation of fundamental algorithms in computational mathematics. The methods and algorithms presented in the course are not in and of themselves used to approximately solve complicated mathematical problems. The algorithms comprise building blocks for complex algorithms that can be used to approximate many mathematical and physical systems. The algorithms that will be covered include basic root finding methods, direct methods for approximate solution of linear systems of equations, matrix operations, eigenvalue and eigenvector computations, polynomial interpolation, numerical differentiation formulas, numerical integration formulas, the Fast Fourier Transform (FFT), along with other concepts related to computational mathematics.

The lion's share of the work in this course will involve the implementation of algorithms using some higher level language (C, C++, Python, Fortran, Java, etc.) along with some analysis. Students will need to know how to apply theorems from a standard calculus course. The theorems will be presented without proof in this course. The main analytic tools you will need to use throughout the course are Taylor series, results from linear algebra related to row reduction, and results for eigenvalue and eigenvector calculations. It will be understandable if you do not quite remember everything when these results are presented. However, you should be able to review the topics from calculus, linear algebra, and other courses when needed. If, in the end, you are still having problems understanding the basic theorems, it will be important for you to talk over the theorems used in the class with me.

The course will involve significant computer programming assignments. Students must know how to write computer code in a high level programming language (e.g; Fortran, C, C++, Java,

and others) or work in a computational platform like Matlab or Maple. If you choose to use Matlab or Maple, you will be required to implement your own versions of the algorithms discussed in class. You can use the intrinsic routines to verify your code. For example, in Matlab there are simple expressions that will prompt the software within Matlab to compute an approximate solution of a linear system of equations. The nuts and bolts are buried deep in the software that Matlab provides. The point of this course is for each student to be able to fully implement algorithms. Using the software provided in Matlab does not allow this. There is another reason why students need to learn all the nuts and bolts that software like Matlab, Mathematica, and Maple cloak. When you are working in a job and need to ship software to your clients, Matlab, Mathematica, and Maple require expensive licensing for their part of the software. These packages are too expensive in most cases. So, you will need to write your own code.

Parallel algorithms for solving problems have been around for a long time. At first, these algorithms were interesting, but maintaining parallel codes can be difficult. In the past decade, the limitations of the physics associated with computing have put limitations on the speed and capacity of a single CPU. Computer companies have started to put more processors into their computers with the idea of improving performance. GPUs have been included to off load the work needed for displaying graphics and the like onto a specialized parallel processing card. Ways to take advantage of multiple cores and CPUs and the GPU onboard your computer have been developed. Students in this course will gain some knowledge of how to use these tools. In particular, concepts from the directive based languages, OpenMP and OpenACC, will be presented to show students the benefits of multicore processing the GPU processing.

Grading: Your grade in the course will be determined by the following:

1. Homework will account for a 40% of a student's grade. Homework must be turned in on time. **Late homework will not be accepted under any circumstances.** If you must be gone, any homework must be turned in before the due date. The only exception to this rule is for a family emergency. Also, codes must be carefully documented. An example of a well documented code will be discussed in class and a version will be posted on line.

2. Two midterms will be given. Each will cover about one half of the content of the course. Each of these midterms will account for 20% or 40% of the grade earned in the course.

3. A portfolio of routines will be required of each student. The portfolio of routines must be written in the form of a software manual. The exact form of the portfolio will be discussed in class. Based on past experience the formatting will be strict in the sense that the entries in the manual will be limited to two pages. The actual format will also be restricted. Software manuals that do not meet the formatting requirements will not be graded. Part of the reason for putting strict requirements for formatting is due to the idea that in a job, you will need to follow the company formatting to have your algorithms included in bigger projects.

If students have questions about assignments and midterms, please contact me. One of the assignments will be to meet with me at least once every two weeks to discuss progress in the course. This means that 2 times each month during the semester you will need to show up at office hours or make an appointment to see me. You will need to do this 8 times during the semester.