ENG ME 700: Computational Mechanics: Nonlinear Analysis and Software Development
Current as of January 20, 2025

Instructor and Class Information

Instructor:
Dr. Emma Lejeune, Assistant Professor of Mechanical Engineering

Office:
730 Commonwealth, EMA 209

Email:
elejeune@bu.edu (please use Piazza for all communications that are not private)

Office Hours:
W – timing TBD, see survey

Class Hours:
MW 10:10-11:55

Classroom:
Lectures will be held in 5 Cummington Mall BRB 122.

Prerequisites:
Mechanics of Materials, Linear Algebra, Differential Equations

Course Website:
Piazza `https://piazza.com/class/m5db1skafd24jz/`, Gradescope

Course Summary

Simulating materials and structures is fundamental to engineering design and analysis. To this end, the field of computational mechanics is concerned with discretizing mathematical models so that they can be solved by a computer. In this course, we will teach both matrix structural analysis and finite element analysis as techniques to solve problems in mechanics with an emphasis on problems with material and geometric nonlinearity. For example, we will cover introductory examples of simulating plasticity and fracture (material nonlinearity), and large deformation and buckling instabilities (geometric nonlinearity). In addition to mastering core concepts in mechanics, students will gain hands-on experience with essential tools and workflows for modern software development. Topics covered will include the use of the command line, Python programming, Markdown for documentation, GitHub for version control and code dissemination, and software

management with Anaconda. The course will also focus on best practices such as Test Driven Development (TDD), Code Coverage, and Code Review. Programming assignments will reinforce theoretical learning and develop practical skills, while an open-ended final project will allow students to apply their knowledge creatively and interact with the open source software ecosystem.

Course Learning Objectives

Technical Knowledge:

- Apply matrix structural analysis methods to solve problems in mechanics, demonstrating understanding of fundamental principles and mathematical formulations
- Implement finite element analysis techniques for solving complex mechanical problems, including proper element selection and mesh refinement strategies
- Analyze and solve problems involving material nonlinearity (plasticity, fracture) and geometric nonlinearity (large deformation, buckling)
- Develop numerical solution strategies for nonlinear problems, including appropriate iteration schemes and convergence criteria

Software Development & Implementation:

- Write, test, and debug Python programs for computational mechanics applications using modern development tools and practices
- Apply test-driven development principles to create reliable and maintainable scientific computing code
- Use version control effectively with Git and GitHub for code management
- Implement proper documentation practices using Markdown and inline code documentation

Professional Skills:

- Evaluate and contribute to open-source scientific software projects in a professional manner
- Review and provide constructive feedback on peers' code through formal code review processes
- Debug and troubleshoot complex numerical implementations using systematic approaches
- Communicate technical concepts and computational results clearly through documentation and presentations

Integration & Application:

- Design and implement comprehensive solutions that integrate mechanical theory with practical software implementation
- Create modular, reusable code that follows software engineering best practices while solving mechanics problems
- Develop and execute verification and validation strategies for computational mechanics implementations
- Complete an independent project that demonstrates mastery of both technical concepts and software development practices

Class Policies

- All students are expected to participate actively during in class code review. This participation will be worth $\approx 25\%$ of the final course grade.
- Students are expected to attend every class, however, if an absence is required for personal or professional reasons there will not be a grade penalty as long as the absence is reported through the "Necessary absences" google form prior to the course meeting and code review is conduced remotely.
- All assignments should be pushed to GitHub prior to the class when they are due. This will facilitate in class code review.
- Code review will be submitted through Gradescope and due at the end of the class period.
- Assignments all contain a "warm-up" assignment and multiple parts.

Academic Misconduct

BU takes academic integrity very seriously. Academic misconduct is conduct by which a student misrepresents his or her academic accomplishments, or impedes other students opportunities of being judged fairly for their academic work. Knowingly allowing others to represent your work as their own is as serious an offense as submitting anothers work as your own. More information on BU's Academic Conduct Code, with examples, may be found at `http://www.bu.edu/academics/policies/academic-conduct-code`

Accommodations for Students with Documented Disabilities

If you are a student with a disability or believe you might have a disability that requires accommodations, requests for accommodations must be made in a timely fashion to Disability & Access Services, 25 Buick St, Suite 300, Boston, MA 02215; 617-353-3658 (Voice/TTY). Students seeking academic accommodations must submit appropriate medical documentation and comply with the established policies and procedures `http://www.bu.edu/disability/accommodations/`

Absence for Religious Reasons

According to Chapter 151C of the General Laws, Commonwealth of Massachusetts, any student in an educational or vocational training institution, other than a religious or denominational educational or vocational training institution, who is unable, because of his or her religious beliefs, to attend classes or to participate in any examination, study, or work requirements on a particular day, shall be excused from any such examination or study or work requirement, and shall be provided with an opportunity to make up such examination, study, or work requirement that may have been missed because of such absence on any particular day. More details can be found here: `https://www.bu.edu/academics/policies/absence-for-religious-reasons/`

Using Generative AI in Coursework

The purpose of this course is to learn new skills. In my personal opinion, the biggest risk of using generative AI tools in this class is that you will deprive yourself of the opportunity to learn new skills in an environment with many resources to support you. That being said, it is completely normal to use any resources that you have access to to figure out how to solve problems while you are coding (e.g., stackoverflow). To this end, the core idealogical tenant of the course generative AI policy is that ultimately you are responsible for the accuracy and validity of the final products that you create, and you are responsible for what you get out of this course.

For english language text generated by AI, please indicate when you have used an AI tool to write any portion of the text (e.g., see examples on course GitHub). If you are unsure about what "counts" in this context, please err on the side of over-reporting generative AI use (e.g., "ChatGPT was used to write a first draft, available here, the final version reported on this page is the result of extensive edits to the generated text"). Again, there is no penalty for using these tools.

For every course assignment with programming, please include in the relevant GitHub repository a text file titled "assignment_#_genAIuse.txt". In the text file, please state what tools were used to help you write code and how you used them. If you did not use any of these tools, please simply state "no AI tools were used to complete this assignment." We are collecting this data for information purposes only, as long as you take full responsibility for the functionality of the code you are allowed (though not necessarily encouraged) to use generative AI tools as a part of your workflow.

Grading

There will be five assignments throughout the semester, each assignment will be broken up into multiple deliverables:

- Assignment 1: Getting setup and solving nonlinear equations 20%
- Assignment 2: Creating your own matrix structural analysis code 20%
- Assignment 3: Creating your own finite element analysis code 20%
- Assignment 4: Using open source finite element analysis software FEniCS 20%
- Final Project: Open ended topic 20%

Approximate grading scheme:

- Code meets basic assignment deliverable – 50%
- Quality of participation in code review – 25%
- Clarity of code and quality of documentation – 25%

Schedule (subject to change, assignment deadlines to be updated)

L01 1/22 Intro to course

L02 1/27 Research Computing Services – crash course, **Highly suggested: get started with setup so you know what questions to ask**

L03 1/29 Newton's Method, **Assignment 1 Warm-up due**

L04 2/03 Intro to Material and Geometric Nonlinearity

L05 2/05 Elasto-Plasticity, **Assignment 1 Part 1 due**

L06 2/10 Matrix Structural Analysis, Introduction

L07 2/13 Matrix Structural Analysis, Software Considerations, **Assignment 1 Part 2 due**

L08 2/18 Matrix Structural Analysis, Geometric Nonlinearity

Note: Tuesday with a Monday schedule

L09 2/19 Matrix Structural Analysis, Geometric Instability

L10 2/24 Finite Element Analysis, Continuum Mechanics Background

L11 2/26 Finite Element Analysis, Numerical Implementation

L12 3/03 Finite Element Analysis, Software Considerations

L13 3/05 Finite Element Analysis, Validation

Spring break is 3/10-3/14

L14 3/17 Finite Element Analysis, Common Pitfalls and Challenges

L15 3/19 Finite Element Analysis, Common Pitfalls and Challenges

L16 3/24 Finite Element Analysis, Wrap Up and Preparation for FEniCS

L17 3/26 Introduction to FEniCS

L18 3/31 Introduction to FEniCS

L19 4/02 Examples of Material Nonlinearity with FEniCS

L20 4/07 Final Project Pitch

L21 4/09 Examples of Material Nonlinearity with FEniCS

L22 4/14 Examples of Geometric Nonlinearity with FEniCS

L23 4/16 Examples of Geometric Nonlinearity with FEniCS

L24 4/21 FEniCS, Wrap Up

L25 4/24 Final Project Presentations and Code Review

L26 4/28 Final Project Presentations and Code Review

L27 4/30 Final Project Presentations and Code Review