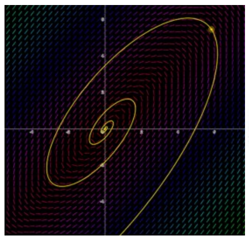
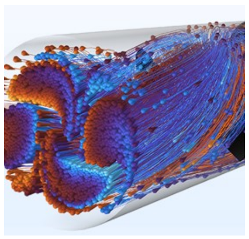
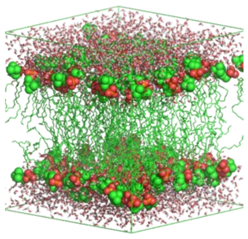
Computational Methods for Physicists and Engineers (ENR 145)



Spring, 2026 (1/14/2026 - 5/8/2026)
For lectures, there are two sections with 18 seats for each:
ENR 145 01 MW 3:00 PM-4:20 PM
ENR 145 02 TR 11:00 AM-12:20 PM
Professor: Xiang Li
Contact Information: xili@coe.edu
Student Hours: walk-in (MW 9:30 am – 12:00 pm MT 1:00 pm – 3:00 pm) or by individual appointments
Department Chair: Firdevs Duru, fduru@coe.edu

Course Description:

It is inconceivable for modern physicists and engineers to plan, calculate, or run anything without the assistance of a computer. Whether doing R&D for a self-driving car or a medical diagnostic device, computer-aided modeling is often the only viable way to make sense of complex systems.

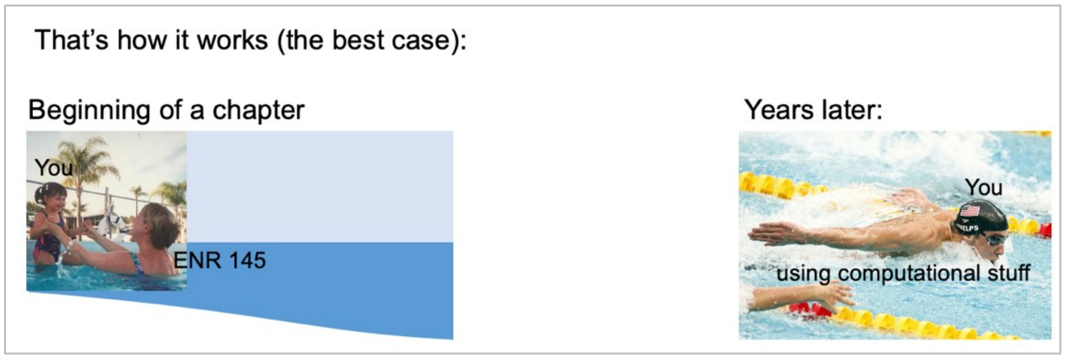
In this course, we will visit some parts of the 'grand computational world.' By the end of the semester, you will dig deep and build projects using the same tools that professionals use. We are not just learning syntax; we are learning how to solve problems.

There is no universal coding tool for everything. In 2026, you cannot do machine learning without Python, handle IT tasks without Linux, or develop embedded systems without C/C++.

Unlike pure Math or CS courses, we focus on **application**. We will cover the *concepts* of math and the *utility* of coding. Ultimately, engineers want to find solutions using existing tools, not re-invent them.

During the course, I will also introduce the non-coder way of doing things like using spreadsheet to visualize and handle data, and ImageJ for image-based data processing.

Below is the illustration of the learning objectives:


That's how it works (the best case):
Beginning of a chapter
You
ENR 145
Years later:
You
using computational stuff

The course was divided by the following chapters:

0) 1 week of pre-computational: numerical methods
0.1) Counting in binary and hexadecimal numbers.
0.2) Boolean operation and truth table reading.
0.3) Intro to encoding, decoding, and error correction.
0.4) Deliverable: hand calculation of Hamming (16,11) extended encoding and error correction.

1) 4 weeks of Python, focusing on:
1.1) General intro to coding (decomposition and iteration).
1.2) Use coding to do a simple maze run.
1.3) Use computational tooling to perform encoding, decoding and error correction for larger data size.
1.4) Deliverable: encoding and error correction with Haming (256, 248) extended for pixel arts.
1.5) Data analysis and data visualization (from ODE solver, system stability visualization to UMAP manifold learning).

2) 4 weeks of COMSOL, focusing on:
2.1) The use of ODE and PDE to explain real-world problems (from pendulum, double pendulum to pendulum with air resistance).
2.2) Crash course of FEA and CFD. (Modeling, meshing, governing equations, boundary conditions, preprocess, convergence, and postprocess)
2.3) COMSOL workshop to build a physics and multi-physics simulation from scratch.

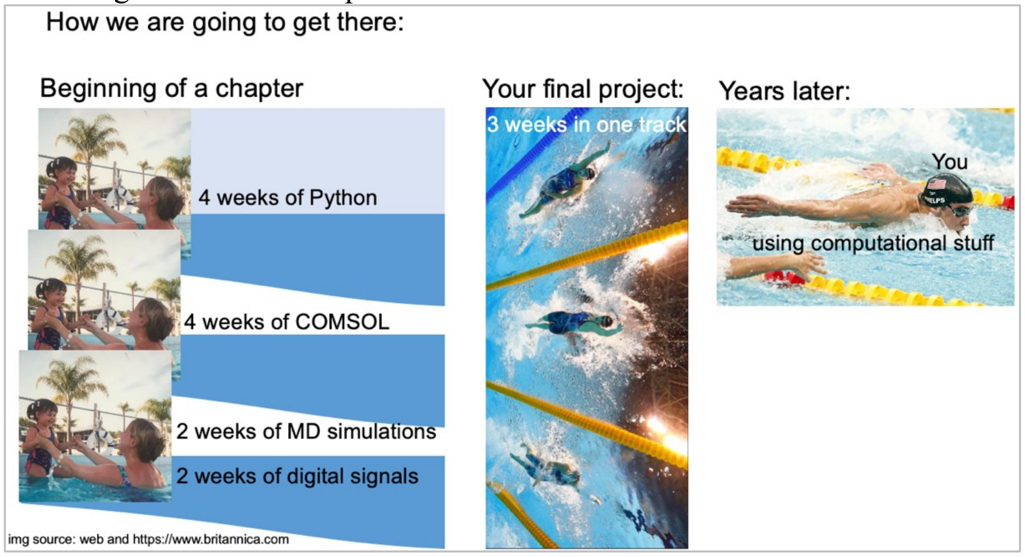
3) 2 weeks of MD simulations, by Prof. Ugur Akgun
3.1) Introduction to Linux systems, remotely accessing computer clusters
3.2) Molecular dynamics simulations in Material Science via LAMMPS
3.3) Molecular dynamics simulations for biological systems via NAMD
3.4) Analysis and visualization of MD simulations

4) 1 weeks of digital toolboxes:
4.1) Data collection and data processing. (digital signal processing and data filtering)
4.2) Workflow of the following essential surviving skills in the new digital age (some modules will be pre-introduced in the previous chapter):

- IDE to make you coding like a pro.

- Prompt and vibe coding with generative AI and Agentic IDE.
- Productivity: Use AI assistants to accelerate administrative and documentation tasks.
- Github page set-up, since everything needs a Github page now.
- Spread sheet, slide decks, design graphs and illustrations for your presentations.
- Image processing with ImageJ.

One crucial component in this course a weeks-long project and demonstrated one's extensive growth in that chapter:


How we are going to get there:
Beginning of a chapter
4 weeks of Python
4 weeks of COMSOL
2 weeks of MD simulations
2 weeks of digital signals
Your final project:
3 weeks in one track
Years later:
You
using computational stuff
img source: web and https://www.britannica.com

Measurable Learning Objectives:

Chapter 0: Mathematical foundations about computational method (1 Week)

0.1) For the mathematical foundations about computational method, the students will:

- Understand and manipulate basic numerical data forms: base 2 and base 16. Familiar with the Hexadecimal coding.
- Understand and implement basic Boolean operations: AND OR NOT NAND NOR and XOR.
- Analyze logic gates with truth tables.
- Understand and implement the very first error correction in the history of coding: Hamming. (ENR325 Fall 2025 course favorite). Hand calculation with 8- and 16-bit extended Hamming coding.
- **The GitHub and IDE setup in the *Digital toolbox (chapter 4)* happens week 1.**

Chapter 1: Python for Data & Algorithms (4 Weeks)

1.1) For introduction to coding, the students will follow the "Karel the Robot" framework from Code in Place (from the Stanford CS department):

- Implement sequential logic with basic coding structure in standard Karel/Python Syntax (move (), turn_left (), and put_beeper ()).
- Use custom functions and parameter inputs to reduce repetition.

- Implement Boolean conditions, and If/Else statement to make decisions. Know the difference between a command and a query.
- Implement iterations (For Loops and While Loops) to achieve automation.
- Combine query and iteration to solve puzzles in a grid system.
- Understand and implement the art of coding: Decomposition and Step-wise Refinement to solve a slightly more challenged maze puzzle in a grid system. The algorithms will be able to adapt to terrain and grid size changes.
- Implement a wall follower, a generalized program that solves any maze.
- Advanced maze solver challenges (1/2): use memory (beepers/colors) as external memory variables.
- Advanced maze solver challenges (2/2): apply logic to detect loops, compare simple reflex and state awareness with examples like islands.
- Use Karel the Robot platform to solve maze: A* search[1] or flood fill[2]. ← Depends on the progress and class feedback, this can be the final project material.

For application with real Python, the students will:

1.2) Based on their prior knowledge in 1.1, the students will implement a Hamming coding algorithm in Python with customed encode () and decode ():
  - Linear algebra basics: vector, matrix dimension (in NumPy), vector matrix multiplication (in NumPy).
  - NumPy basics: bitwise manipulation, shapes and reshaping, broadcasting, and (pro version of) parity check.
  - For Hamming coding: generator matrix, parity check matrix, and vectorized operations. Compare the pros and cons between natural mapping method and systematic mapping.
  - Scaling up: apply the systematic approach to handle larger data sets (256 bits). Boolean indexing, concatenation, decode by searching, and evaluation of speed.
  - Implement a real-life (sound, graph or even video) I/O solution based on the hamming coding, or other error correction codes[3]. ← Depends on the progress and class feedback, this can be the final project material.

1.3) This section provides tools to visualize differential equations in python. The students will implement a vector space to simulate damped pendulum.
  - Math background: derivative, kinematics of a damped pendulum system, and the phase space.
  - Coding basics: The implementation of NumPy grids, SciPy ODE solver, and Matplotlib as a plotting tool.
  - Time series of a damped pendulum model: implement ODE equations, and use phase space and animation to visualize the friction and system stability.
  - Implement manifold learning, reduce high dimensional data to 2D space with the help of modern-day data science (UMAP).

[1] https://en.wikipedia.org/wiki/A*_search_algorithm
[2] https://en.wikipedia.org/wiki/Flood_fill
[3] https://en.wikipedia.org/wiki/Error_correction_code

Chapter 2: Multi-Physics Simulation with COMSOL (4 Weeks). Student will be able to:

- Understand the concept of translate real-world physical phenomena into mathematical models using Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs).
- Be familiar and implement the workflow of COMOSL simulation: Pre-processing, Meshing, Solving, and Post-processing.
- Be familiar with building and importing a CAD design into COMSOL.
- Construct a simulation of a dynamic device (pendulum) and a fluidic system (air flow and damping) from scratch in COMSOL.
- Analyze simulation convergence and mesh quality to ensure result validity.
- Implement the workflow of decomposition and gradual iteration to realize a multi-physics simulation coupling at least two physical domains.

Chapter 3: Molecular Dynamics (MD) Simulations by Professor Ugur Akgun (2 Weeks)

- Establish a secure remote connection to a supercomputing cluster and manage data via the Linux terminal.
- Execute production-level MD simulations for both crystalline materials (LAMMPS) and solvated proteins (NAMD).
- Troubleshoot common simulation errors, such as "lost atoms" or energy minimization failures.
- Extract quantitative physical properties from trajectory files using command-line tools and scripting.
- Communicate molecular-level phenomena through high-quality 3D animations and statistical plots.

Chapter 4: Digital toolboxes (1 Week):

- Set-up GitHub pages for personal and course projects. (week 1)
- Install and implement modern IDE to code like a pro. (week 1)
- Knows how to prompt coding and use agentic AI to speed up coding and documentation workflow.
- Knows how to use agentic AI tools to learn and practice class contents.
- Be familiar with office and communication tools such as spread sheet, slides, graph and illustrating, and image processing tools.

Required Texts and/or Course Materials:

- Lecture slides and resources will be posted and maintained on Moodle and GitHub.
- Free online coding courses and recommended schedules will be posted on Moodle and Github.
- There is no traditional textbook; curated online references and readings will be provided throughout the term.
- The links to recommended book chapters and tutorials will be provided on Moodle and GitHub.

Course Policies:

Official catalog description: Computational Physics Methods for Physicists & Engineers
Designed to be a survey of the computational methods used by physicists. Students learn basics

of Linux systems, Python, Jupyter Notebook, Github, shell scripting, and data analysis tools, such as MATLAB and ROOT. The students will learn and apply simulation tools and electronic interfaces to their design projects. This course is project-based and requires intensive individual participation in prototyping.

Schedule of major due dates: Homework typically due one week later it's assigned. ***Pre-lab assignments*** will due at the beginning of each class. Project report will be due before the start of the new chapter. Final project will be presented as a slide deck and live tech demo. Here's the list of major due dates:

Week	**Major due:**	**Chapter**
1	Demo	0
3	Demo	1
5	Presentation	1
6	Demo	2
9	Presentation	2
11	Report	3
12	Report	4
13	Proposal	Final project
14	Tech demo	Final project
15	Presentation	Final project

The individual types of assignments are listed as follows:

Assignment	**Basis for grading**	**What will be recorded**
Foundation skills	Overall correctness	Success/Retry
Applications	Completeness and overall correctness	Exemplary/Success/Retry
Reports	Completeness, overall correctness, writing, and presentation	Exemplary/Success/Retry/or Incomplete
Advanced skills	Completeness and efforts	Exemplary/Pass/No pass
Presentation	Writing, documentation, and presentation	Pass/No pass

Method for determining final grade: The Learning Loop

In this course, we mirror the professional engineering world. Your goal is not to be perfect on the first try, but to iterate.

- 0 (Retry): The work is incomplete or incorrect. You must revise and resubmit.
- 1 (Success): The work meets the quality standard.
- Exemplary: The work exceeds expectations (e.g., exceptional documentation, novel solution). Note: "Exemplary" marks can help boost your grade in borderline cases.

Based on the nature of each section (concepts, math, coding, debugging), we will have tasks in the form of

- Homework (typically due one week later, can be anything from work out bugs to literature review and report).
- Pre-lab assignments (read some pages, learn and practice some (fun) numerical skills before the class).
- In-class pop quiz (work out a math problem with pen and paper, or debugging a code with all the help you can get.
- In-class real-time demo (build a project/server/notebook and bring it online).

The distribution of tasks in each chapter is listed as follows (excluding the final project and presentation, which can happen in any chapter), the frequence and the compositions are subject to change based on the pace and feedbacks as the class progress.

Chapter	Foundation	Demo	Reports	Advanced skills	Presentation
0	4	1			
1	5	3	1	3	2
2	2	1	1	1	1
3		1	1		
4	3	1	1	1	1
Total:	14	7	4	5	4

Your grade is assigned with the table listed below. Each row indicates the minimum number of successful results needed to satisfy the requirement for that grade. For example, you will be required to earn 10 or more success mark on foundation skills to get grade of B.

Grade	Foundation skills	Demo	Lab reports	Advanced skills	Presentation
A	12	5	3	3	3
B	10	4	3	2	2
C	8	3	2	1	1
D	5	2	1	0	0

A grade of "F" is assigned if not all of the requirement for a "D" are met.
Plus grades: A "plus" is given on a letter grade if you satisfy all the requirements plus one of the following:

- More than half of the recorded grades are exemplary, when applicable.
- Complete two categories in the grade table for the next grade level up, no including presentation.

Minus grades: A minus grade is assigned if you meet the criteria for Foundation/Demo/Report but miss the target for Advanced/Presentation by exactly one point.

Participation expectations, if applicable: **Participation:** Active participation in class is the most efficient way to learn. It saves you hours of self-teaching later. **Attendance:** Attendance is tracked but not graded. I will follow up with students who have excessive absences, but no direct penalty will be incurred. You do not need to seek permission to miss a class.

Late work policy, if applicable:
You will start the semester with 5 tokens. You will spend 1 token to:

- Extend the deadline on any submissions by 36 hours.
- Retry a second time (i.e., submission for a third time).

Acceptable methods for submitting work: Microsoft PowerPoint and Google Slides are the preferred submission formats via Moodle. **Handwritten Notes:** These are acceptable only if they are clearly legible, but they cannot earn an 'Exemplary' grade. Building high-quality slide decks is a critical career skill—now is the time to practice!

Course management system /website, if applicable: Lecture slides and resources will be posted and maintained on Moodle and GitHub.

Information about Final Exam and course meetings through the end of the term: A crucial component of your final project is the presentation. It's not just about the data and the code; it's about **storytelling**. You will be surprised by how much effort it takes to tell a good technical story.

Any additional information or class policies on ChatGPT/AI/generative technology:

General rule-of-thumb: AI tools are roller skates, not crutches.

AI tools (like ChatGPT, Gemini, and Copilot, etc.) may be used as long as they support your learning and do not replace your independent thought or effort.
Allowed: Using AI to explain error messages, generate boilerplate code, or brainstorm ideas.
Prohibited: Copy-pasting code you do not understand or cannot explain.
Citation rule: You must append an "AI Appendix" to any assignment where AI was used.

- Code: Wrap AI-generated blocks in comments:
  ```
  ### Begin AI Code ###
          AI code here
  ### End AI Code ###
  ```
- Images: Caption clearly with examples like [Image generated by XXX generator].
- Prompt Log: Include the specific prompts you used at the end of your submission as supplementary log.