MMAE 450 — Computational Mechanics II

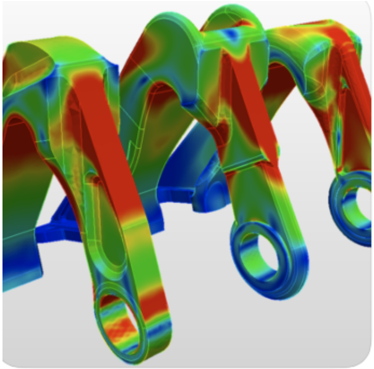
Day 1: Course Introduction & Computational Workflow

Mike Gosz

Spring 2026

Why Computational Mechanics?

- Many engineering problems have **no closed-form solution**
- Computation bridges **theory** and **real systems**
- Central in:
  - heat transfer
  - solid mechanics and dynamics
  - transport phenomena

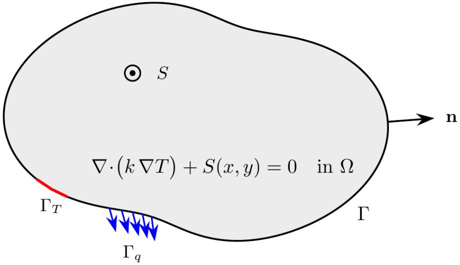


Stress distribution in mechanical part with complex geometry.

What Is Computational Mechanics?

1. Physical system (assumptions + model)
2. Governing equations
3. Numerical approximation (discretization)
4. Computational solution
5. Interpretation and validation

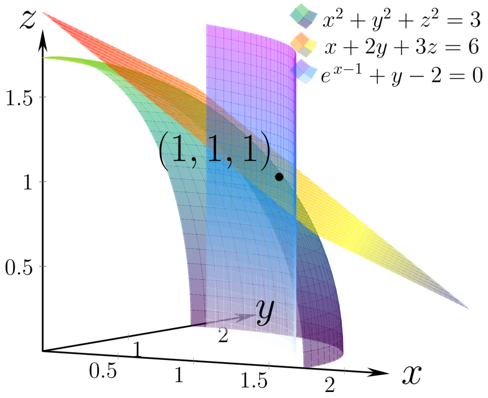
Key message: Computational mechanics is about **models**, not just numbers.



Complex geometry $\Rightarrow$ discretization $\Rightarrow$ computation

Where This Course Fits

- Builds on:
  - calculus
  - differential equations
  - linear algebra
- Develops:
  - numerical thinking
  - computational implementation
  - interpretation of results
- Prepares you for:
  - research and modeling
  - engineering simulation tools
  - machine learning models and cloud workflows
  - data analytics



Intersection of nonlinear surfaces representing a system of equations.

What This Course Emphasizes

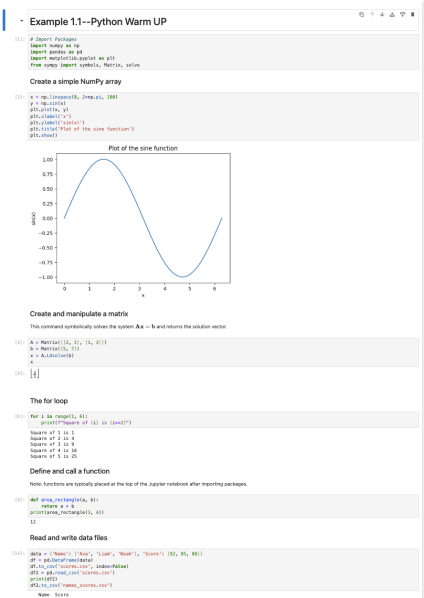
- Modeling, not memorization
- Algorithms, not button-pushing
- Understanding:
  - accuracy
  - stability
  - limitations and failure modes

We will care why methods work—and when they fail.

Tools We Will Use

- **Python** as the computational language
- **Jupyter notebooks** for code + math + narrative
- Core libraries:
  - NumPy (arrays, linear algebra)
  - SymPy (symbolic math, derivations)
  - Matplotlib (plots and visualization)
  - SciPy (numerical solvers, eigenproblems)
  - Pandas (data handling and post-processing)
  - scikit-learn (regression and surrogate models)
  - PyTorch (physics-informed neural networks)
  - Cloud workflows for scalable computation

We will briefly review Python and notebooks to ensure everyone is aligned.

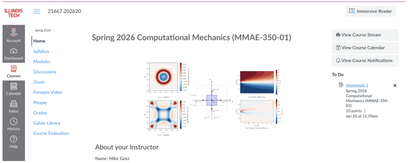


Why Jupyter Notebooks?

- Combine: formatted text, equations, executable code, and plots
- Ideal for:
  - exploration
  - verification of derivations
  - clear communication of results

How the Course Is Structured

- Weekly modules on Canvas
- Mix of:
  - lectures
  - guided notebooks
  - homework assignments
- Two midterms + final exam or final project



A consistent weekly rhythm: learn $\rightarrow$ practice $\rightarrow$ assess

What You Should Do This Week

- Review the syllabus (PDF)
- Complete Module 0 setup (Python + environment)
- Run Notebook 01 and make small edits
- Start Homework 1 (reflection + small computations)