

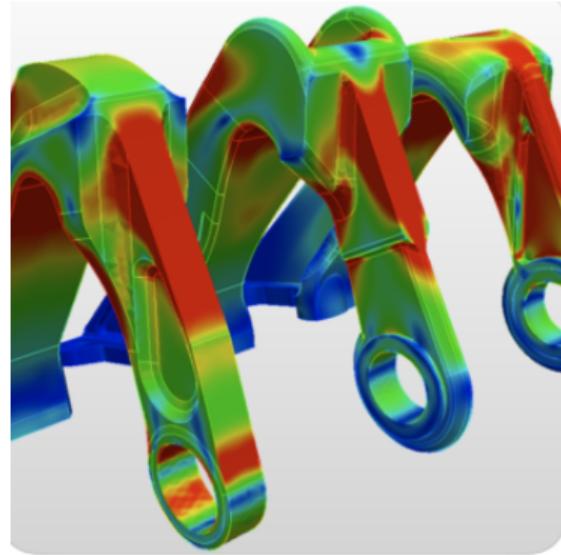
MMAE 450 — Computational Mechanics II

Day 1: Course Introduction & Computational Workflow

Mike Gosz

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

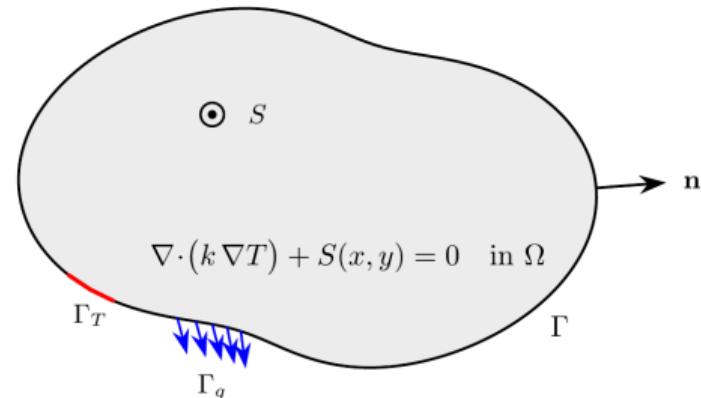


Stress distribution in mechanical part with complex geometry.

What Is Computational Mechanics?

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

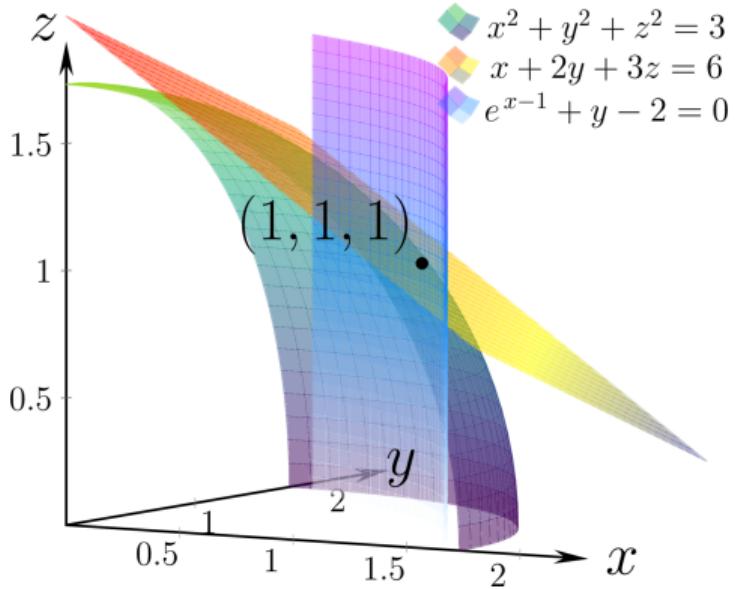
Key message: Computational mechanics is about **models**, not just numbers.



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.

Example 1.1--Python Warm UP

```
# Import Packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sympy import symbols, Matrix, solve
```

Create a simple NumPy array

```
(1): x = np.linspace(0, 2*np.pi, 1000)
y = np.sin(x)
plt.plot(x, y)
plt.title('A')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.suptitle('Plot of the sine function')
plt.show()
```

Plot of the sine function

Create and manipulate a matrix

```
This command symbolically solves the system  $\mathbf{Ax} = \mathbf{b}$  and returns the solution vector.
```

```
(4): A = Matrix([[2, 1], [1, 3]])
b = Matrix([1, 1])
x = A.LUsolve(b)
x
```

```
(4): [1]
```

The for loop

```
(5): for i in range(1, 6):
    print(f"Square of {i} is {i**2}")
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
```

Define and call a function

```
Note: functions are typically placed at the top of the Jupyter notebook after importing packages.
```

```
(6): def area_rectangle(a, b):
    return a * b
print(area_rectangle(3, 4))
```

```
12
```

Read and write data files

```
(14): data = {'Name': 'Max', 'Class': 'Math', 'Score': 78}, {'Name': 'Sam', 'Class': 'Math', 'Score': 85}, {'Name': 'Liam', 'Class': 'Math', 'Score': 90}, {'Name': 'Oscar', 'Class': 'Math', 'Score': 80}, {'Name': 'Ethan', 'Class': 'Math', 'Score': 75}, {'Name': 'Noah', 'Class': 'Math', 'Score': 82}, {'Name': 'Aiden', 'Class': 'Math', 'Score': 79}, {'Name': 'Caleb', 'Class': 'Math', 'Score': 87}, {'Name': 'Benjamin', 'Class': 'Math', 'Score': 83}, {'Name': 'Daniel', 'Class': 'Math', 'Score': 81}, {'Name': 'Jacob', 'Class': 'Math', 'Score': 86}, {"Name": "Max", "Score": 78}, {"Name": "Sam", "Score": 85}, {"Name": "Liam", "Score": 90}, {"Name": "Oscar", "Score": 80}, {"Name": "Ethan", "Score": 75}, {"Name": "Noah", "Score": 82}, {"Name": "Aiden", "Score": 79}, {"Name": "Caleb", "Score": 87}, {"Name": "Benjamin", "Score": 83}, {"Name": "Daniel", "Score": 81}, {"Name": "Jacob", "Score": 86}
```

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

The screenshot shows the Illinois Tech Canvas course page for Spring 2026 Computational Mechanics (MMAE-350-01). The course navigation bar includes links for Home, Syllabus, Modules, Discussions, Zoom, Precept Videos, People, Grades, Grade Library, and Course Evaluation. The main content area displays course details and several thumbnail images representing course content.

A consistent weekly rhythm: learn → practice → 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)