MMAE 350 — Computational Mechanics

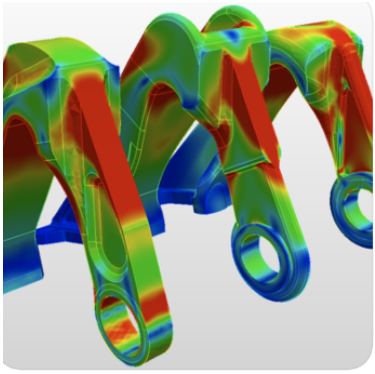
Day 1: Course Introduction & Computational Workflow

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

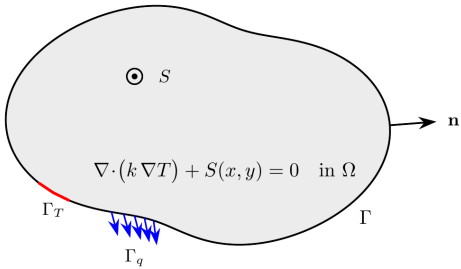


Stress distribution in mechanical part with complex geometry.

What Is Computational Mechanics?

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

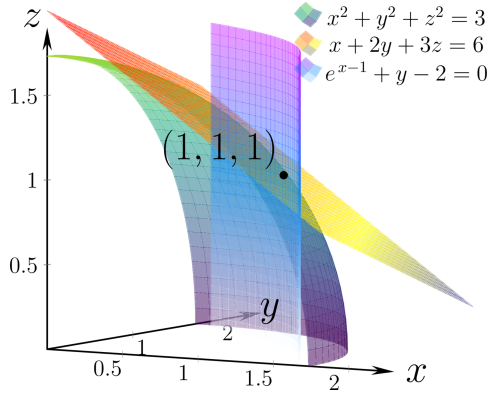
Key message: Computational mechanics is about **models**, not just numbers.



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:
  - upper-level MMAE courses
  - research and modeling
  - engineering simulation tools
  - machine learning
  - data analytics



Intersection of nonlinear surfaces representing a system of equations.

What This Course Emphasizes

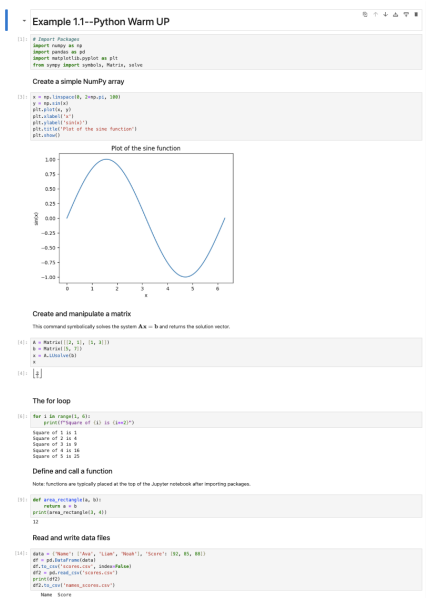
- Modeling, not memorization
- Algorithms, not button-pushing
- Understanding:
  - accuracy
  - stability
  - limitations and failure modes

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)
  - Matplotlib (plots)

No prior Python experience is required.



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

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('Plot of the sine function')
plt.show()
```


Create and manipulate a matrix

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

```
A = Matrix([[2, 1], [1, 3]])
b = Matrix([5, 7])
x = A.inv()*b
x
```

```
⎡8/5⎤
⎢   ⎥
⎣9/5⎦
```

The for loop

```
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.

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

```
12
```

Read and write data files

```
data = {'Name': ['Ava', 'Liam', 'Noah'], 'Score': [92, 85, 88]}
df = pd.DataFrame(data)
df.to_csv('scores.csv', index=False)
df2 = pd.read_csv('scores.csv')
print(df2)
df2.to_csv('names_scores.csv')
```

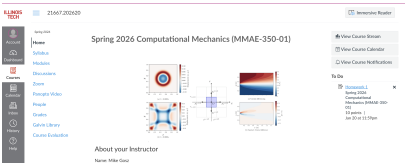
```
  Name  Score
```

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)