You can access these slides on the course Github:
https://github.com/natrask/ENM1050

ENGR 1050
Intro to Scientific Computation

Lecture 01 – Computing hype day and course logistics

Prof. Nat Trask
Mechanical Engineering & Applied Mechanics
University of Pennsylvania

What is this class about?

Scientific Computation

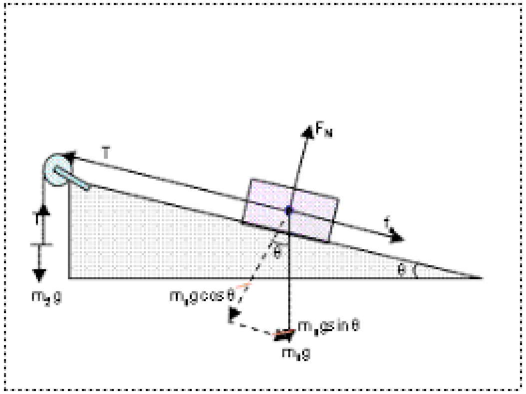
Solving a technical problem   using a computer

Our objectives:

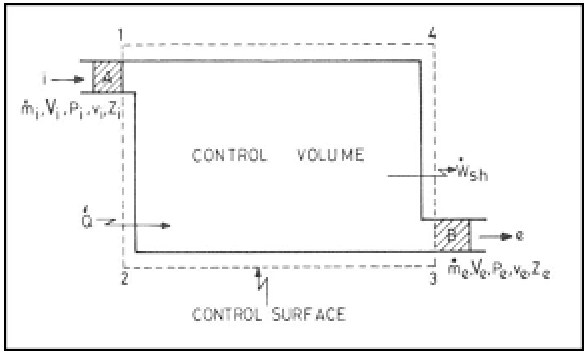
- Translate an English description of a technical problem into a computational model
- Choose a numerical method for solving that problem
- Programming basics to code and debug your approach
- Visualize data and interpret results

Learn where computational techniques fit into your toolbox as an engineer

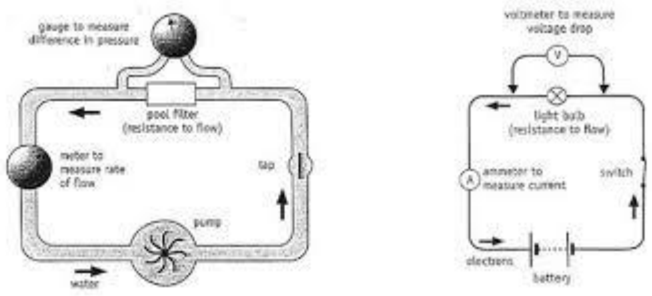
What is scientific computation?



Free body diagrams



Control volume analysis



Circuit models

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0$$

$$\rho\left[\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}u+\frac{\partial u}{\partial y}v+\frac{\partial u}{\partial z}w\right]=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+\rho g_x$$

$$\rho\left[\frac{\partial v}{\partial t}+\frac{\partial v}{\partial x}u+\frac{\partial v}{\partial y}v+\frac{\partial v}{\partial z}w\right]=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)+\rho g_y$$

$$\rho\left[\frac{\partial w}{\partial t}+\frac{\partial w}{\partial x}u+\frac{\partial w}{\partial y}v+\frac{\partial w}{\partial z}w\right]=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)+\rho g_z$$

Differential equation models

Over the course of your degree, you will learn analytic tools to model systems

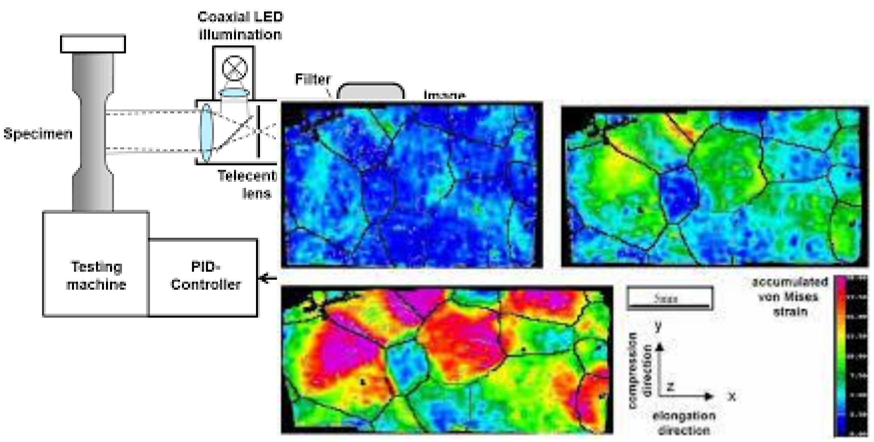
What is scientific computation?

Analytic models are

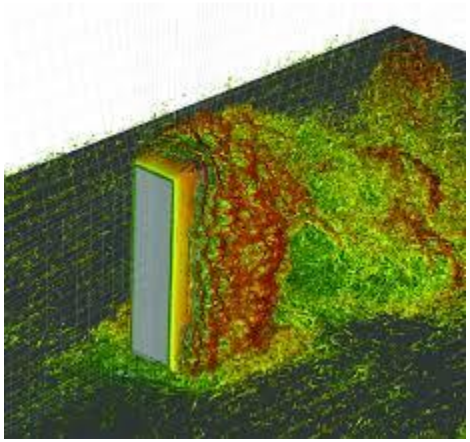
Good at:
Giving closed form expressions
Interpretability
Exploring qualitative regimes

Bad at:
Handling complexity
Multiple scales
Multiple physics
Limited by the math we can work out by hand!

When you can’t work a problem by hand, you turn to:



Experimental methods



Computational methods

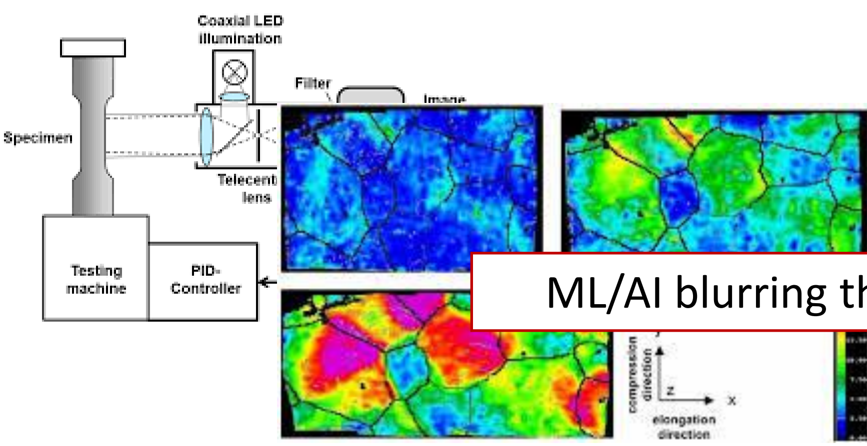
What is scientific computation?

Analytic models are

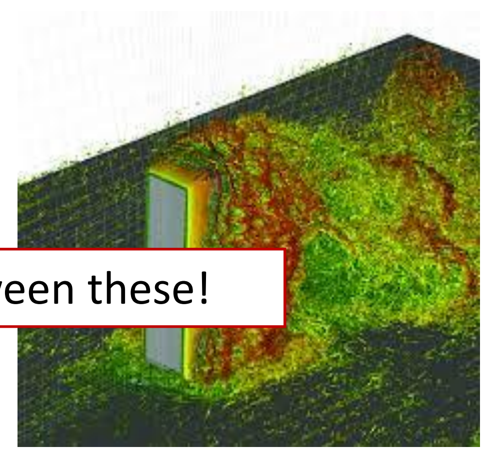
Good at:
Giving closed form expressions
Interpretability
Exploring qualitative regimes

Bad at:
Handling complexity
Multiple scales
Multiple physics
Limited by the math we can work out by hand!

When you can't work a problem by hand, you turn to:



ML/AI blurring the lines between these!



Experimental methods

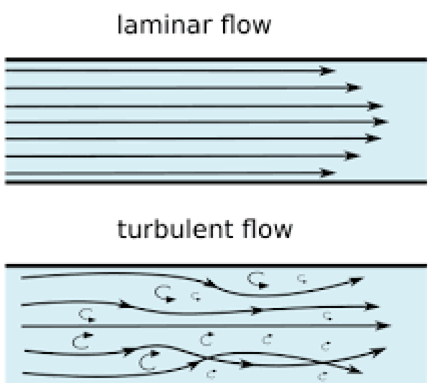
Computational methods

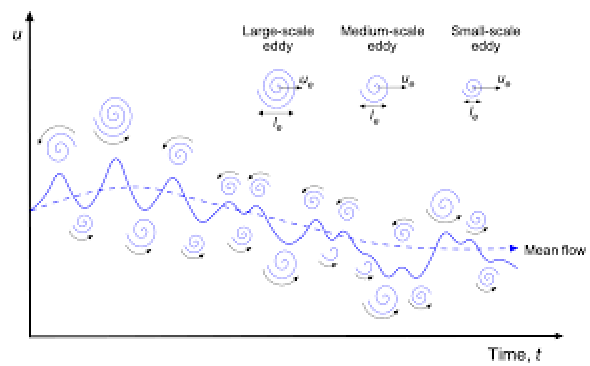
What are the limits of scientific computation?

1 FLOP = 1 floating point operation

An add, subtract, multiply, divide

FLOPS are currency – and scales with the number of unknowns you're going to solve for






PHYSICAL REVIEW FLUIDS

Highlights Recent Accepted Collections Authors Referees Search Press

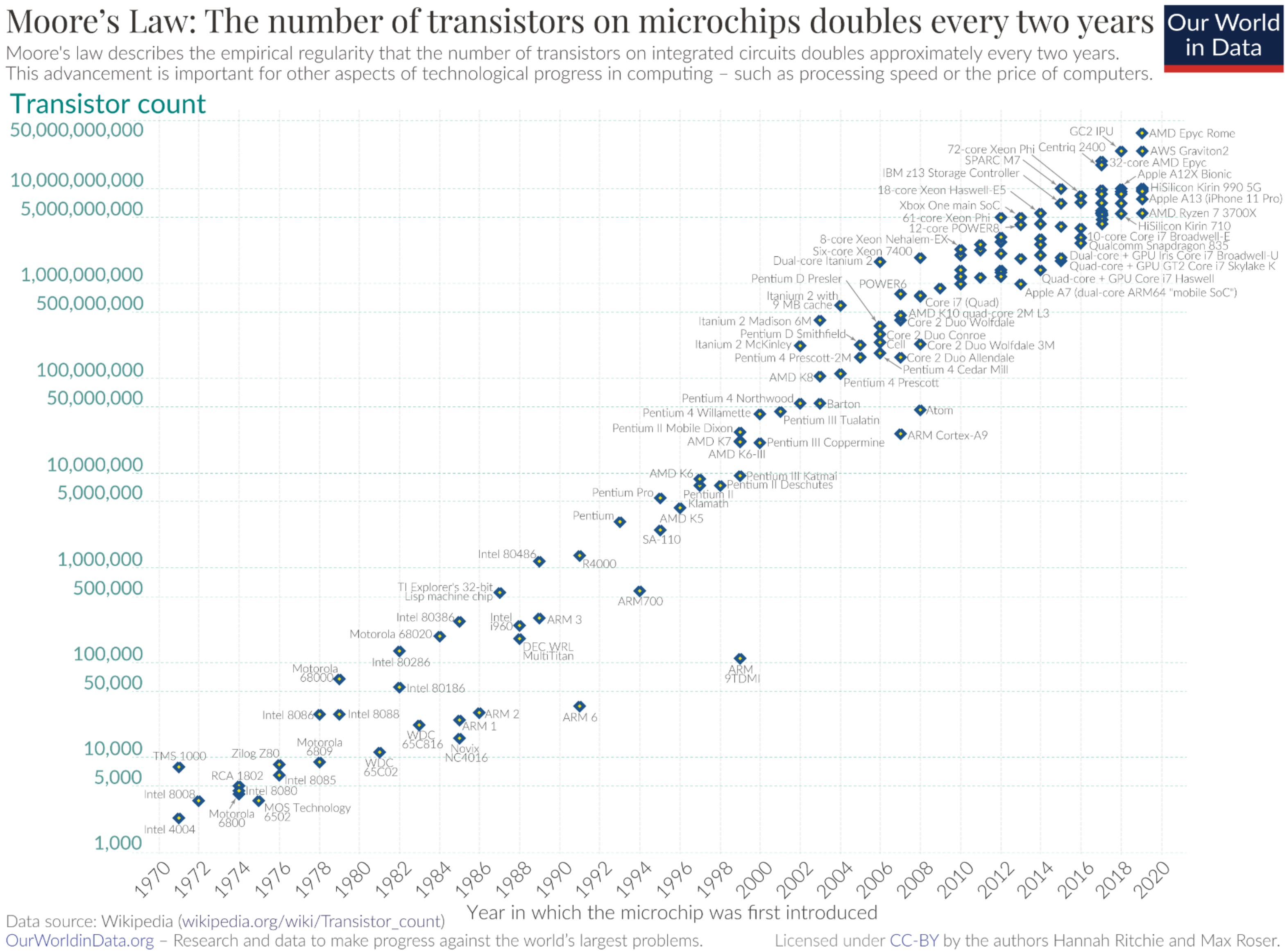
Degrees of freedom and the dynamics of fully developed turbulence

Diego Donzis and Shilpa Sajeev

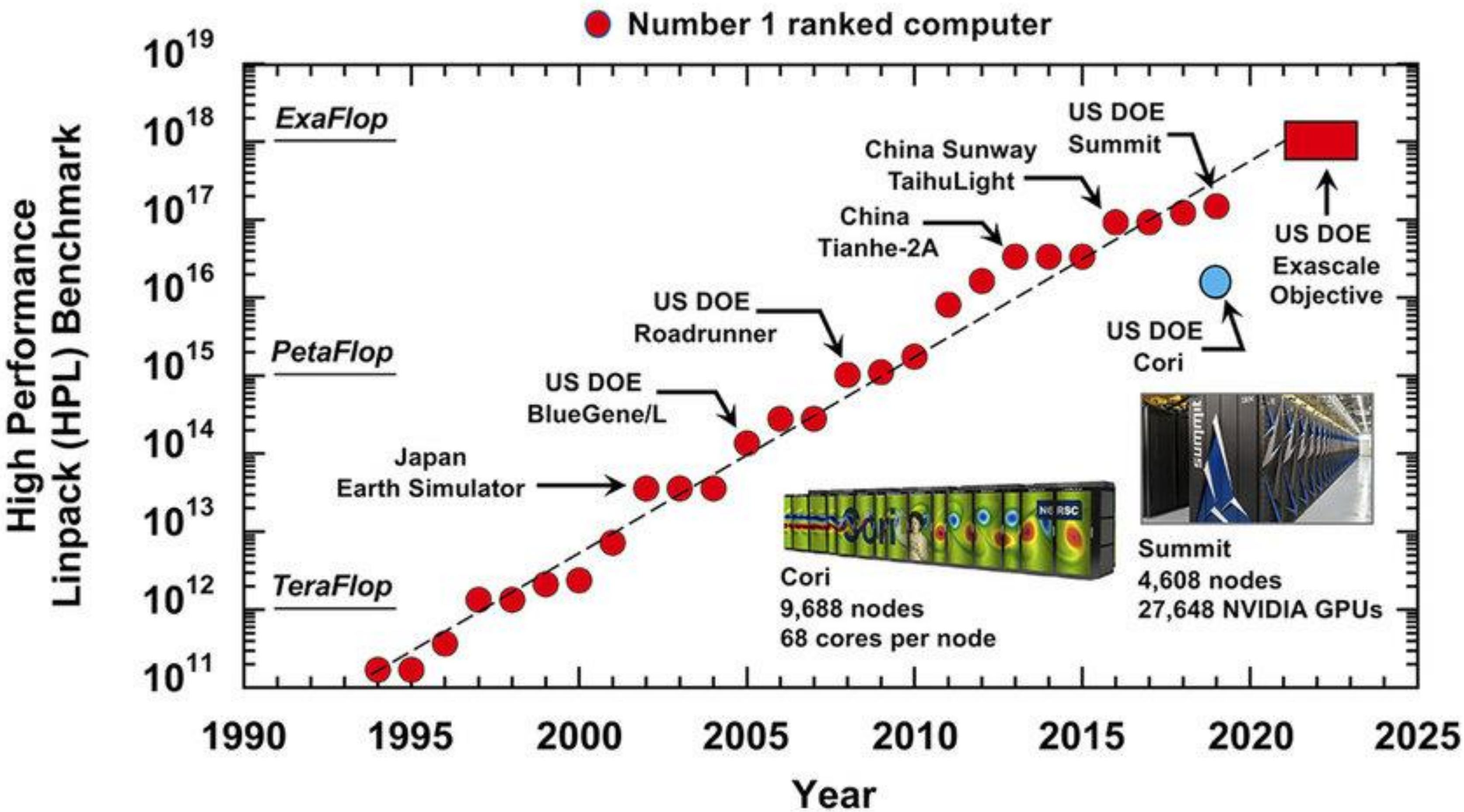
Phys. Rev. Fluids **9**, 044605 – Published 15 April 2024

What's the biggest problems we can solve right now?

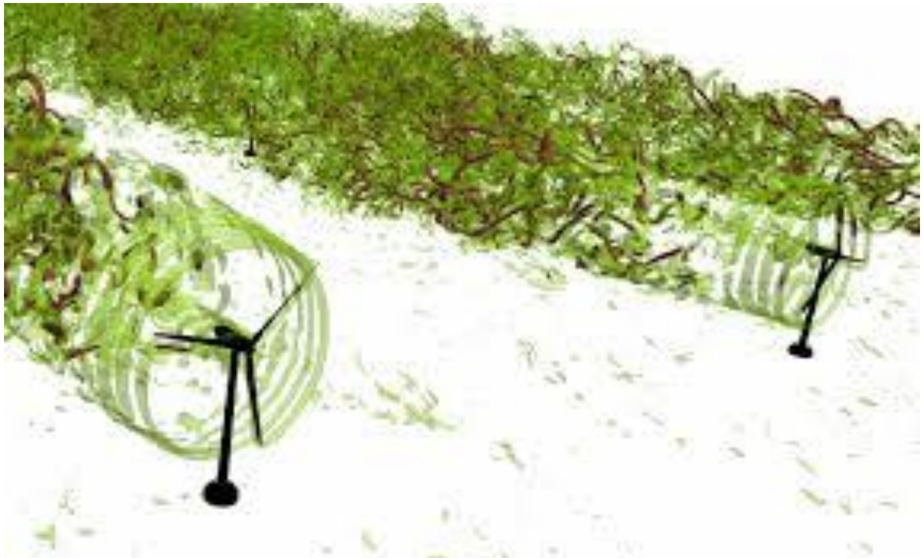
What are the limits of scientific computation?

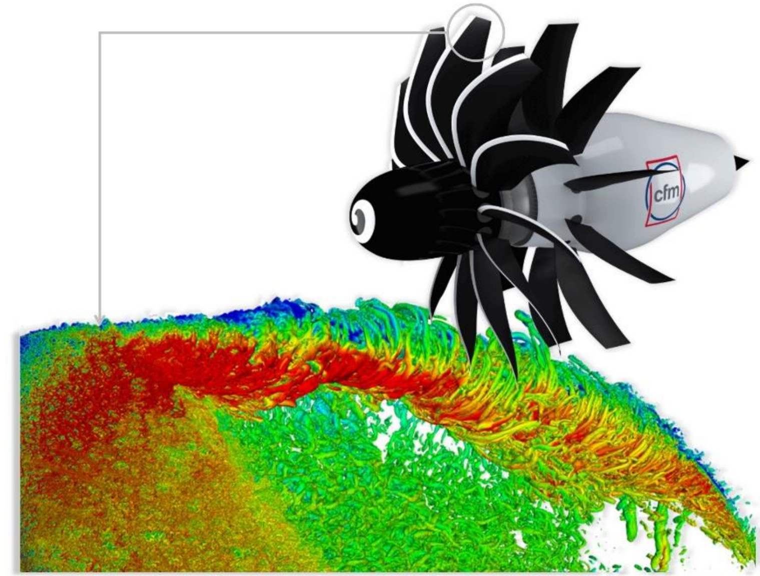


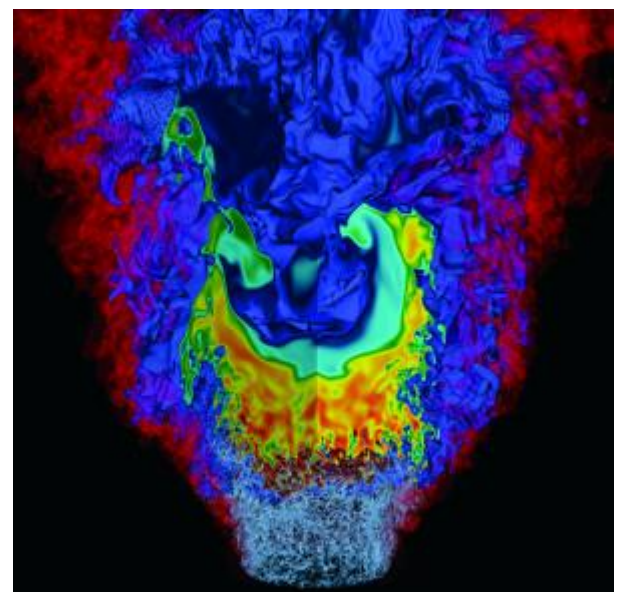
What are the limits of scientific computation?



What are the limits of scientific computation?



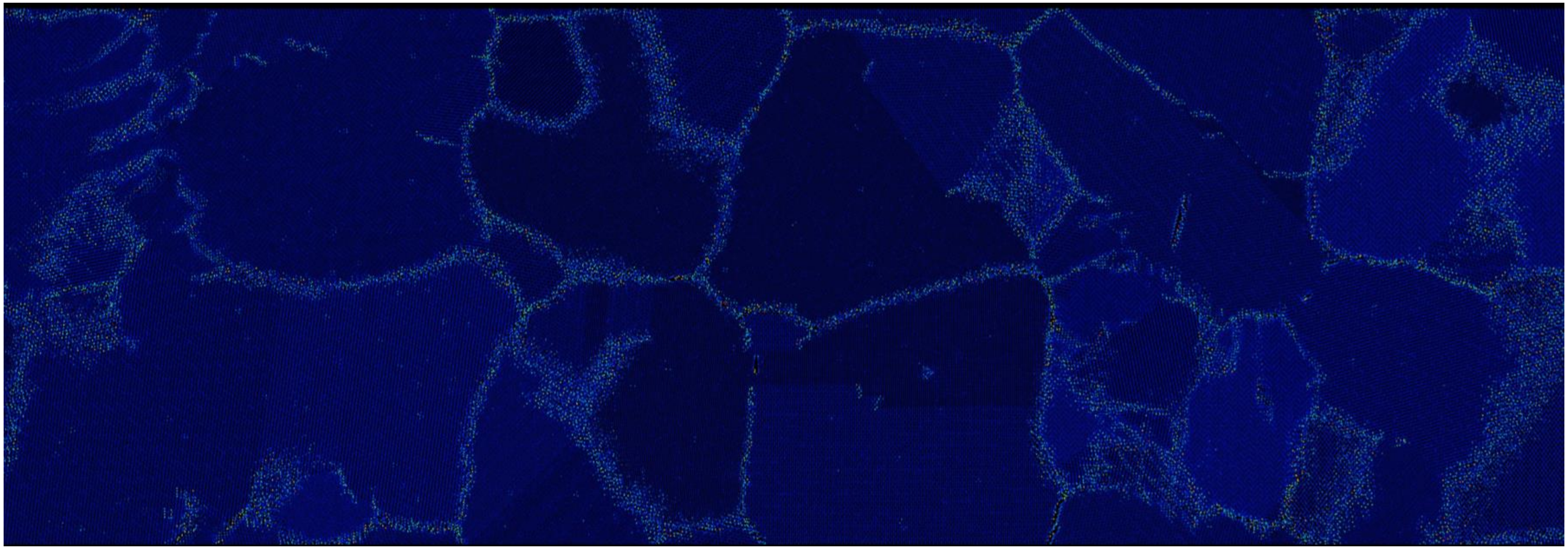


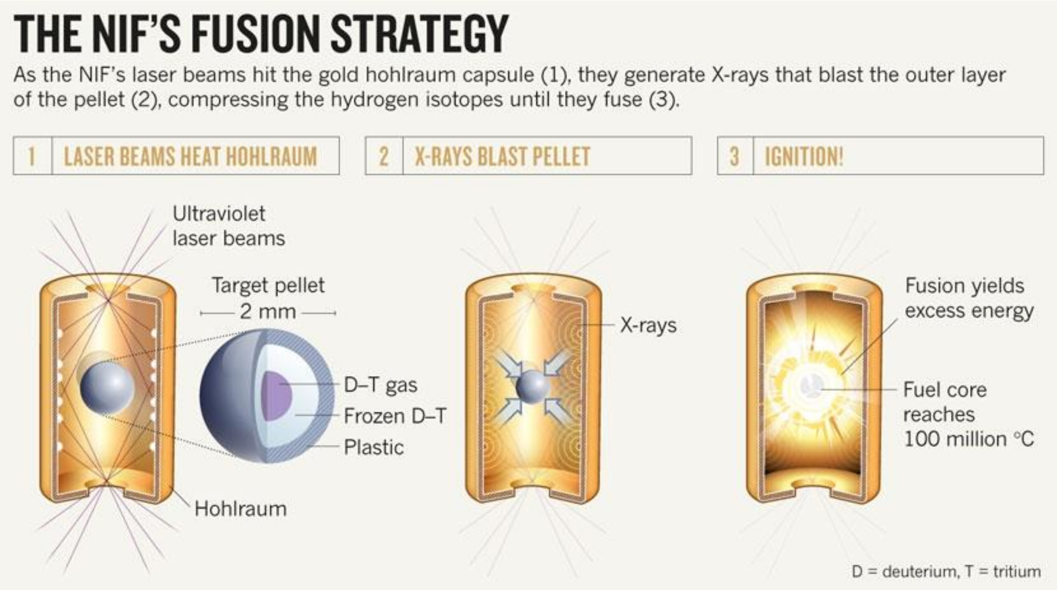


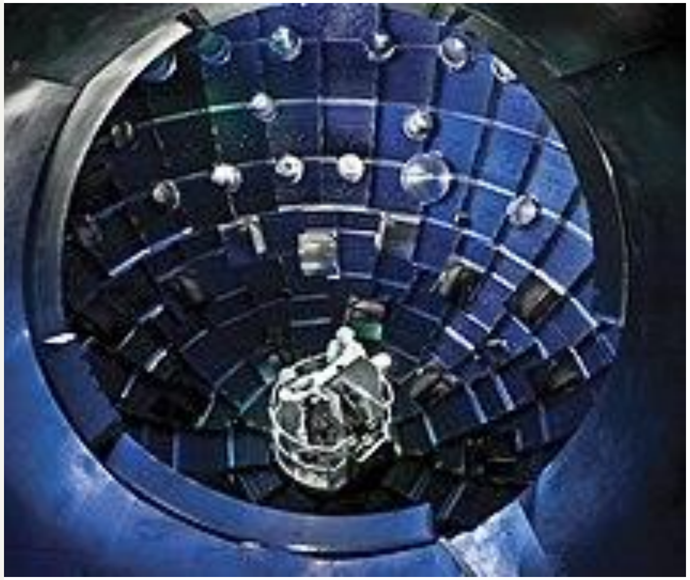
These are the worlds largest simulations, using an obscene amount of energy (an exascale computer consumes the energy of 16000 homes!!!!)

<u>We cannot brute force computational solutions to practical engineering problems</u>

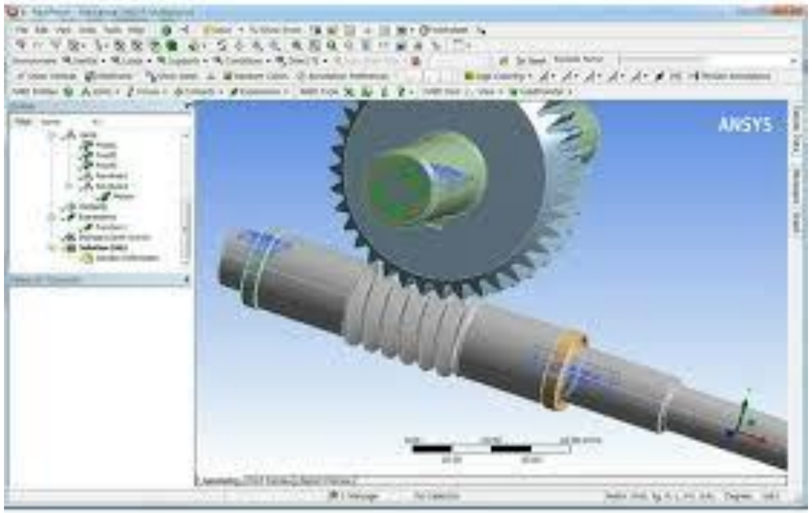
Fusion power at National Ignition Facility powered by exascale simulations (Lawrence Livermore National Lab)

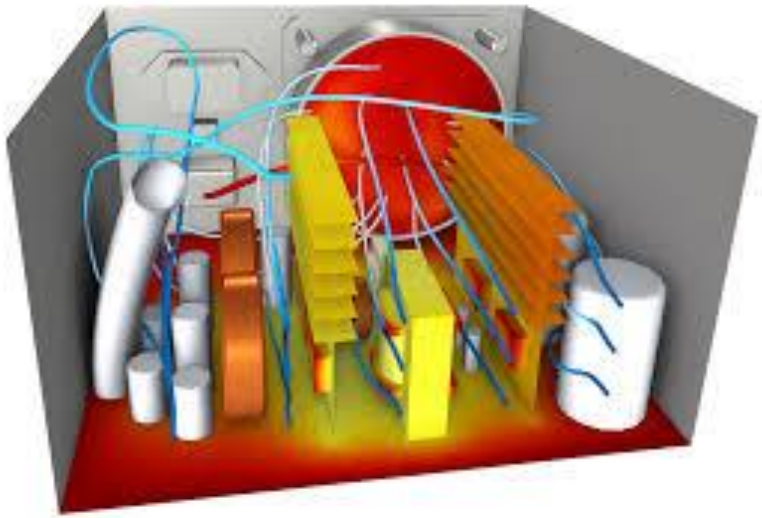






Limits of practical scientific calculation?





Ansys

COMSOL MULTIPHYSICS®

Preprocessing (CAD) → Calculation (FEM/CFD) → Post-processing

Who am I?

Dr. Nat Trask

Associate Professor

Mechanical Engineering & Applied Mechanics

Secondaries in AMCS, Materials

Joint faculty appointment Sandia National Labs

You can call me: Nat, Prof Trask, Dr Trask

I like listening to/playing music, baseball, and am sleep deprived from small children

Who am I?

I joined Penn faculty in August 2023

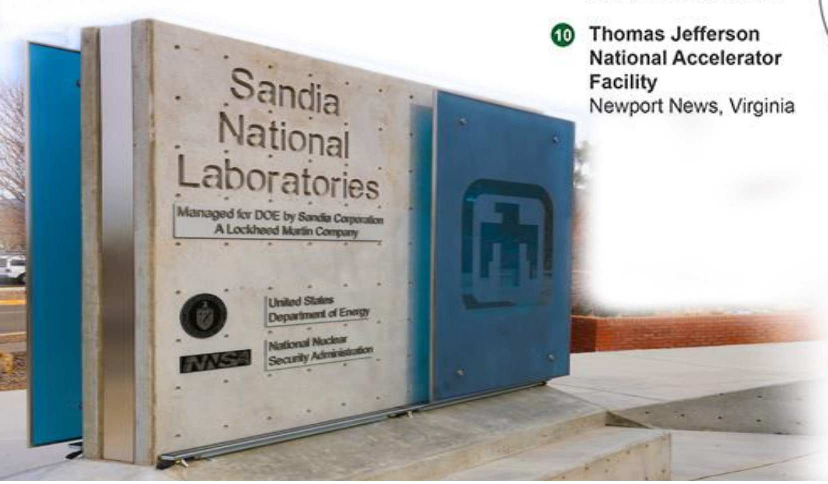
I was senior staff at Sandia National Laboratories 2016-2023, and hold a joint faculty appointment still

I earned my Ph.D. in Applied Mathematics at Brown University in 2015, and hold dual a dual Bachelor degree in Mechanical Engineering and Applied Mathematics from University of Massachusetts.

I've run a consulting simulation business since 2010.

My primary research interests are:
Simulating problems that are too complicated to derive models for with pencil and paper – fusion power, shock physics, semiconductors, material discovery

Department of Energy and the National Lab Complex



Office of Science Laboratories

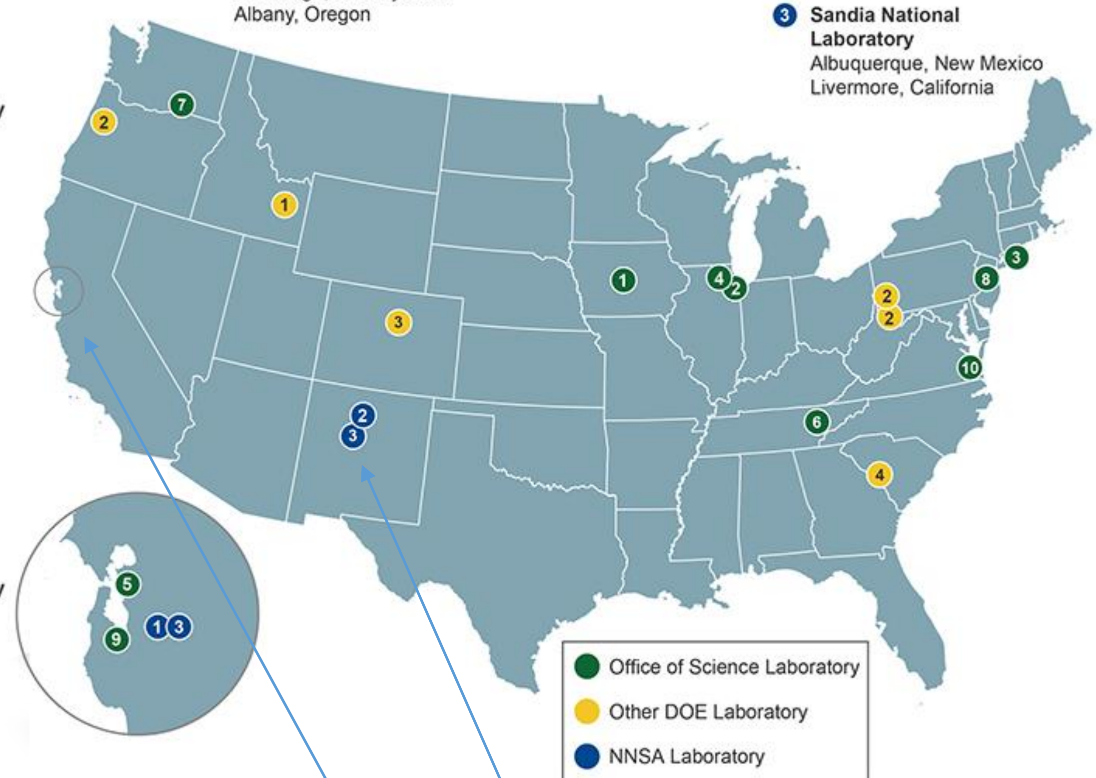
1. **Ames Laboratory**
   Ames, Iowa
2. **Argonne National Laboratory**
   Argonne, Illinois
3. **Brookhaven National Laboratory**
   Upton, New York
4. **Fermi National Accelerator Laboratory**
   Batavia, Illinois
5. **Lawrence Berkeley National Laboratory**
   Berkeley, California
6. **Oak Ridge National Laboratory**
   Oak Ridge, Tennessee
7. **Pacific Northwest National Laboratory**
   Richland, Washington
8. **Princeton Plasma Physics Laboratory**
   Princeton, New Jersey
9. **SLAC National Accelerator Laboratory**
   Menlo Park, California
10. **Thomas Jefferson National Accelerator Facility**
    Newport News, Virginia

Other DOE Laboratories

1. **Idaho National Laboratory**
   Idaho Falls, Idaho
2. **National Energy Technology Laboratory**
   Morgantown, West Virginia
   Pittsburgh, Pennsylvania
   Albany, Oregon
3. **National Renewable Energy Laboratory**
   Golden, Colorado
4. **Savannah River National Laboratory**
   Aiken, South Carolina

NNSA Laboratories

1. **Lawrence Livermore National Laboratory**
   Livermore, California
2. **Los Alamos National Laboratory**
   Los Alamos, New Mexico
3. **Sandia National Laboratory**
   Albuquerque, New Mexico
   Livermore, California



Sandia Albuquerque

Sandia Livermore

What will computation look like when you go on the job market?

time

Building and solving models

AI/ML

?????
Quantum? Chips in our brains?
?????

Solution: Strong fundamentals in concepts of computing and math

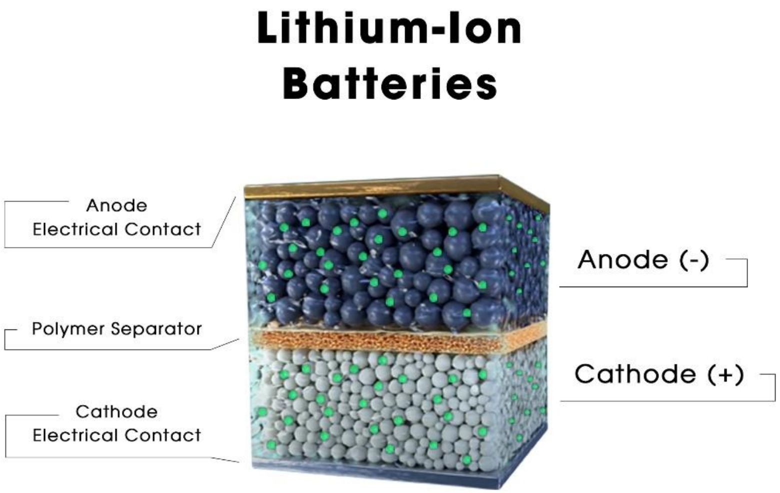
In this course will learn
a little introduction to:

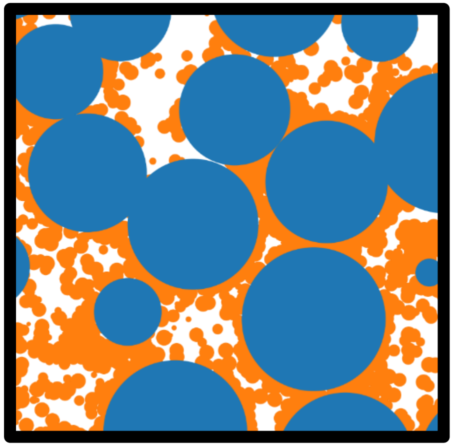
Incorporating data into physics models

Machine learning

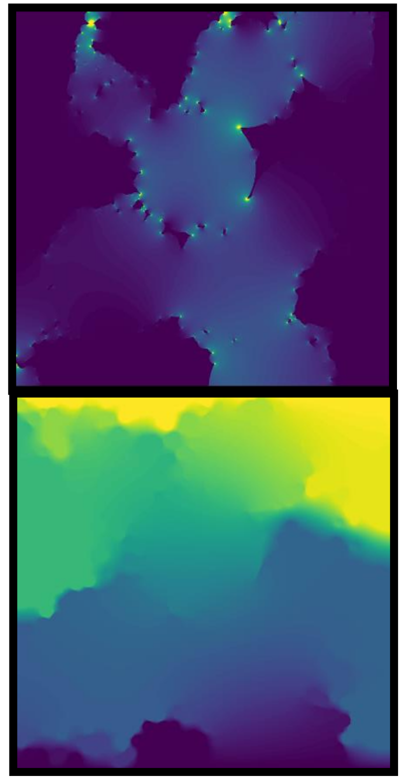
Some examples from my own research

How can we be clever with really expensive simulations?

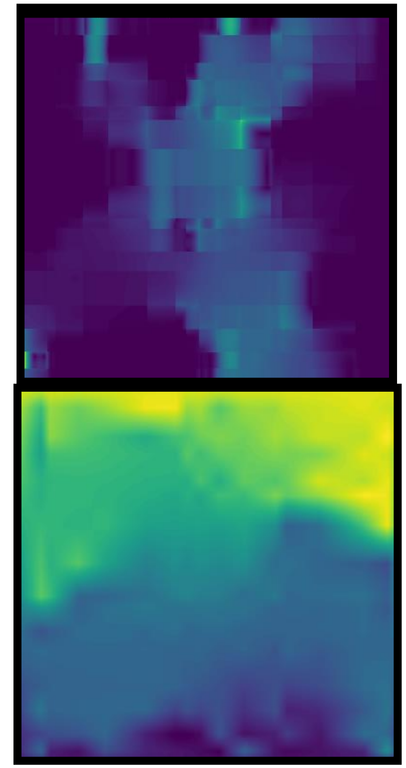




CT-Scan of Lithium ion battery microstructure



FEM simulation on as-built microstructure



Machine learn a cheaper model that preserves **transport pathways**

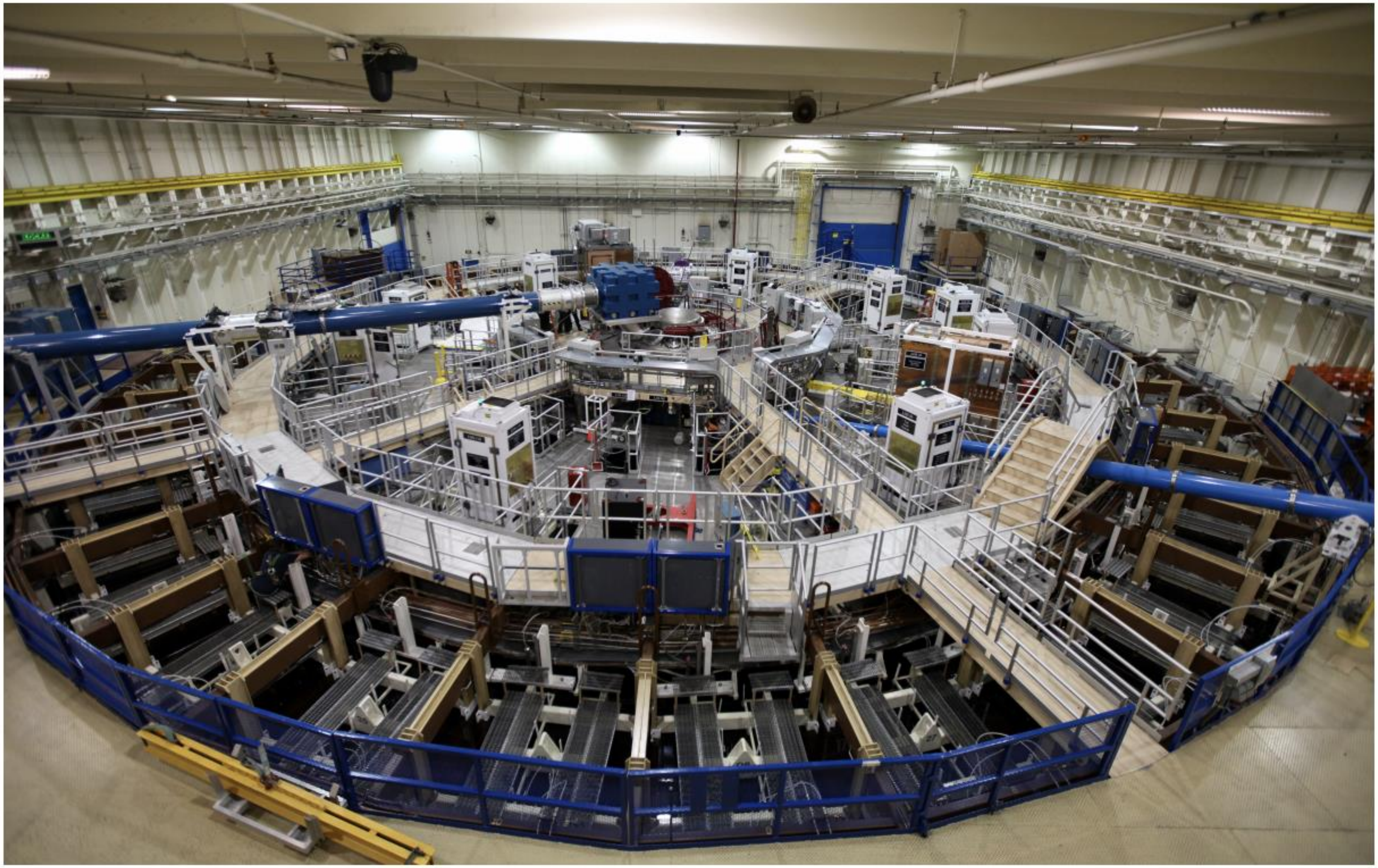
In this course will learn
a little introduction to:

Incorporating data into physics models

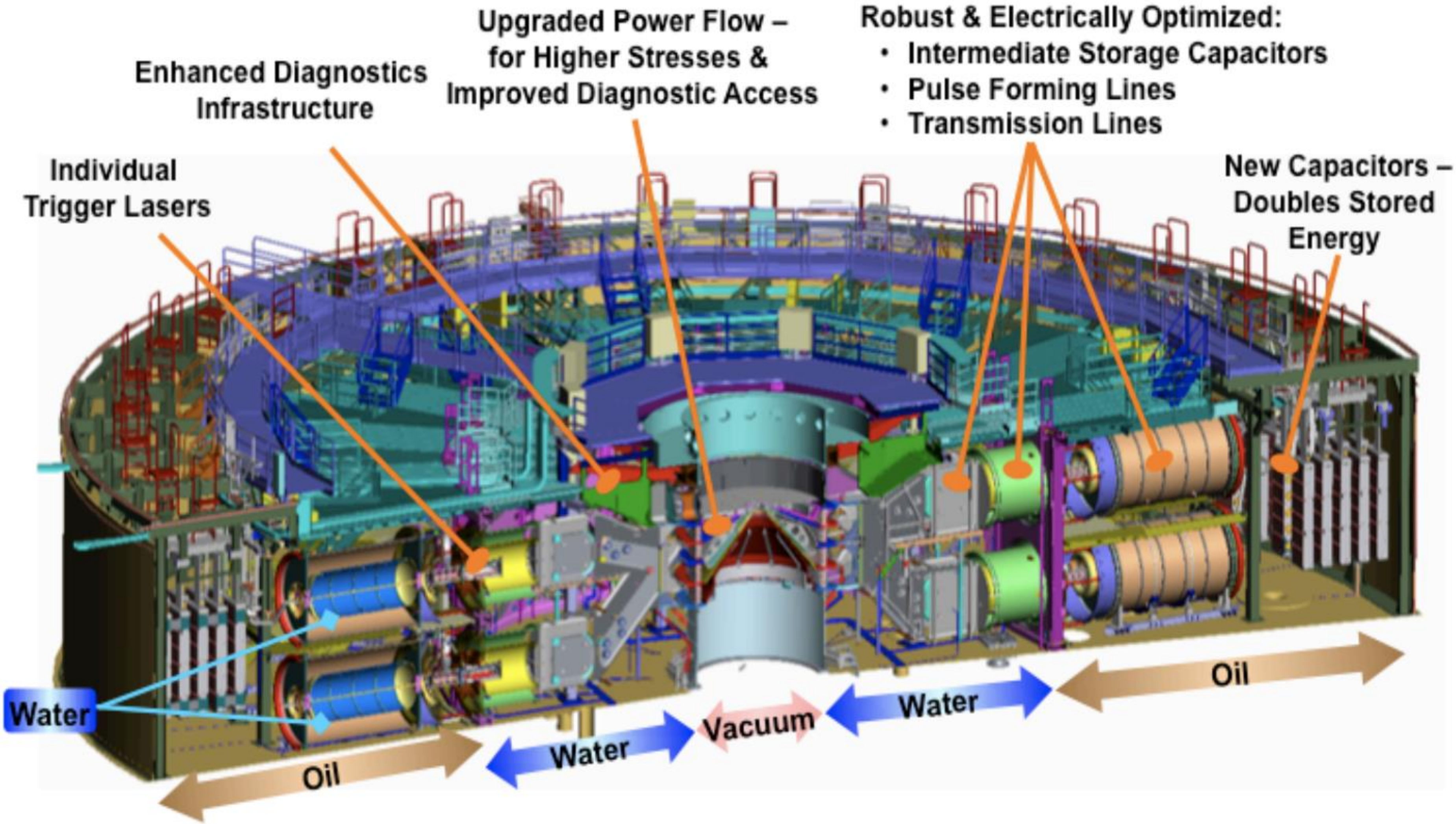
Machine learning

Some examples from my own research

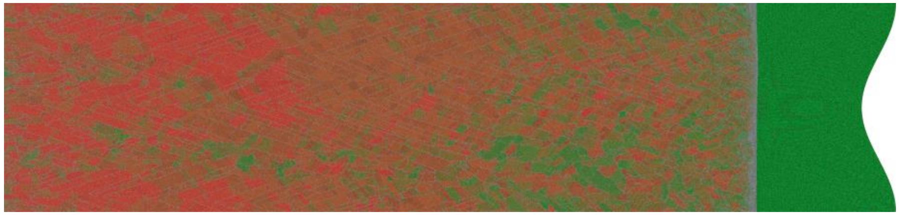
High energy experiments on Z-machine



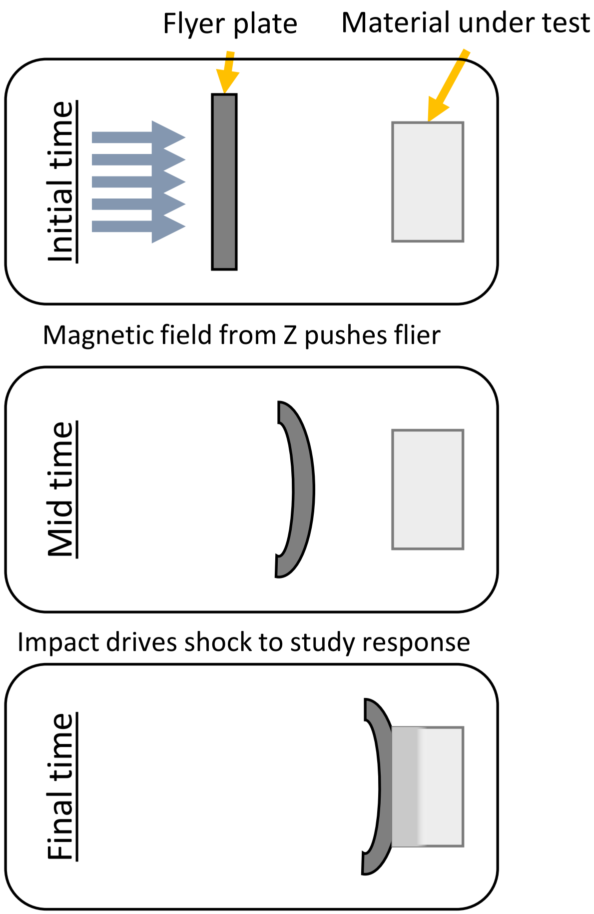
A very big capacitor bank

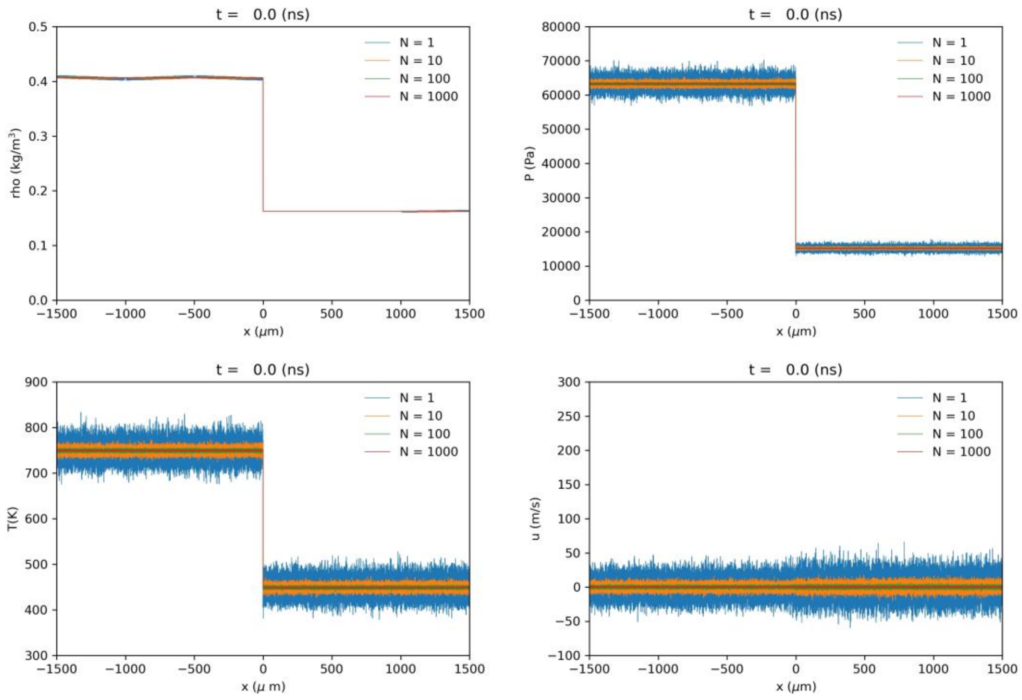


Blurry lines between analytics, experiments and models



Simulated data: Molecular simulations of every single atom





In this course will learn a little introduction to:

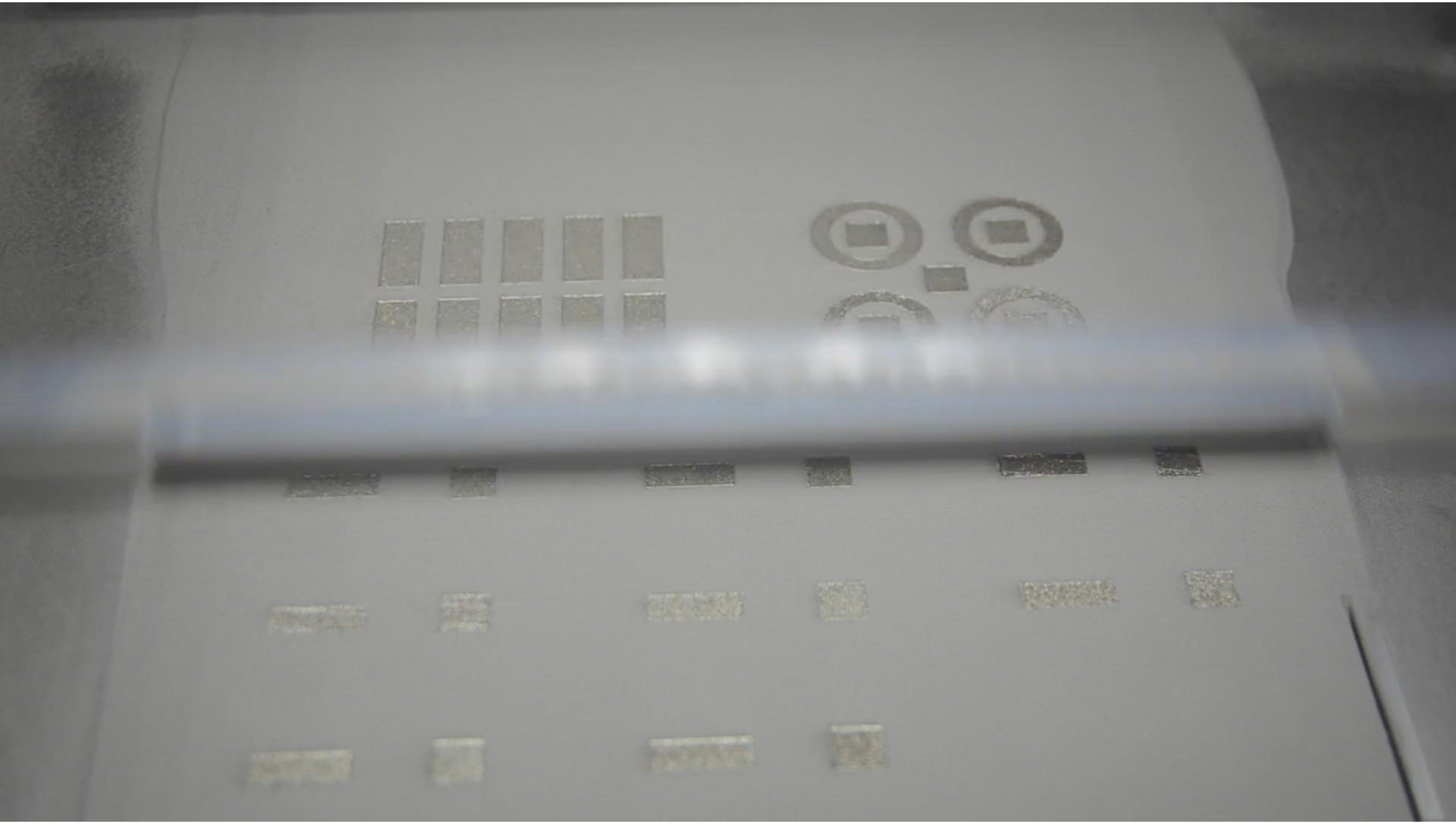
Incorporating data into physics models

Machine learning

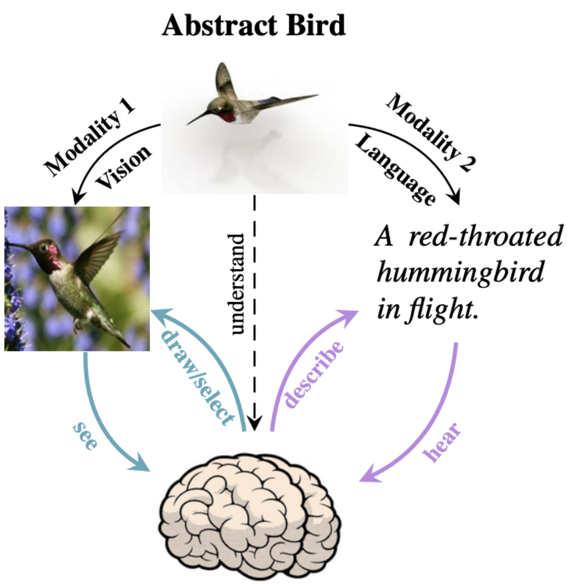
Some examples from my own research

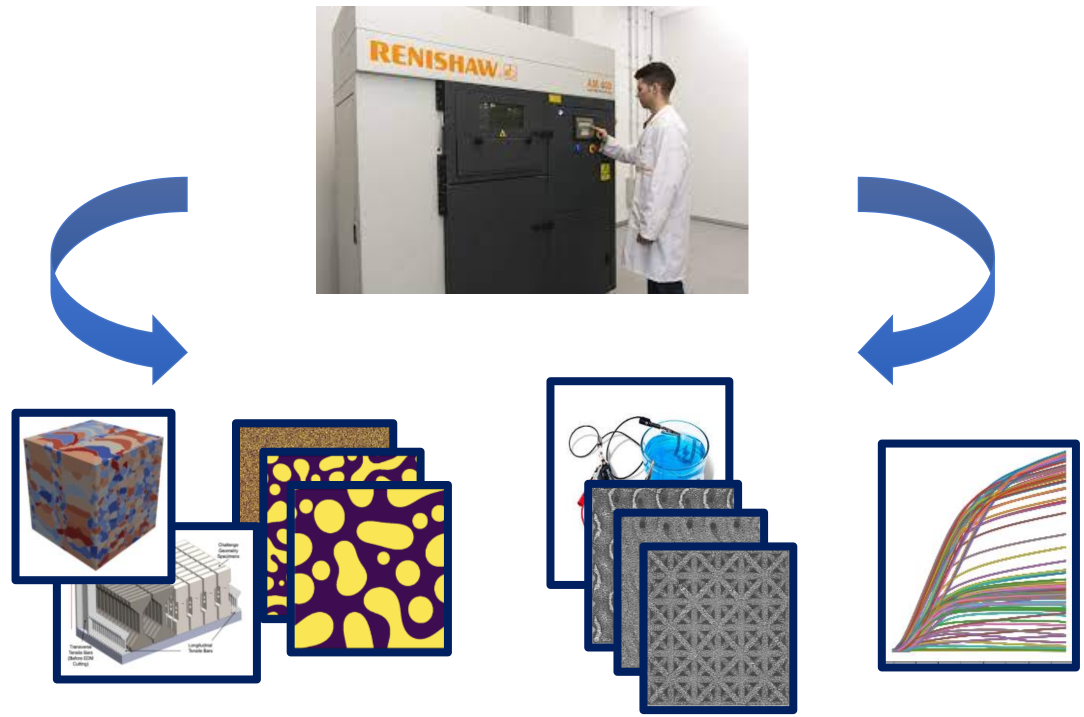
Advanced manufacturing processes

Multimodality: heterogeneous data
Process parameters: laser, powder, speed
High-fidelity characterization: microscopy, XRF, XRD, TEM, SEM
Low-fidelity signals: light, sound, images, profilometry



What do we mean by multimodal deep learning?





Traditional Multimodality

Assimilate heterogeneous data sources so they're greater than sum of their parts

e.g. audio + text to generate automatic subtitles

"Physics agnostic" Modalities

Classical material characterization

e.g. microscopy (physical), direct simulation (synthetic)

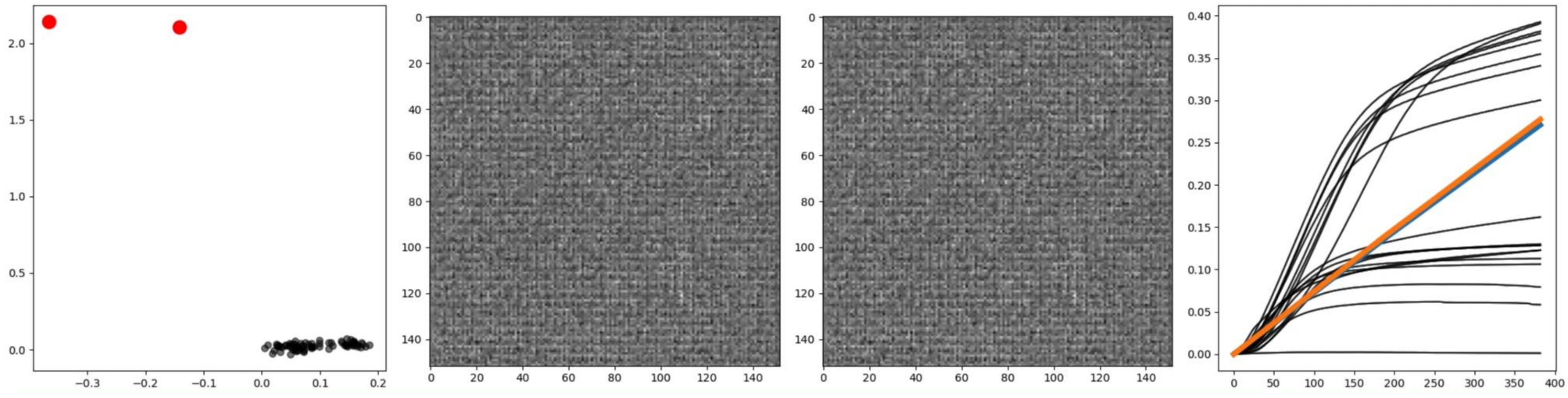
"Physics-amenable" Modalities

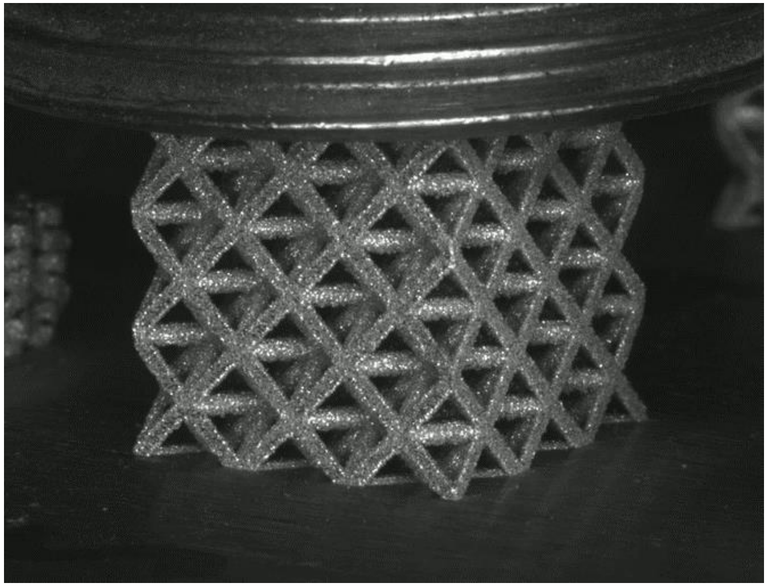
Unconventional measurements

e.g. spectroscopy, electrical signals, sound, photographs

Shi, Y., Siddharth, N., Paige, B. and Torr, P.H., 2019. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *arXiv preprint arXiv:1911.03393*.

Physics-based disentanglement of metal AM





1. Images of lattices and stress/strain curves are disentangled in latent space
2. Once disentangled, images are calibrated to clusters
3. Once disentangled, expert models calibrate into two populations
4. **Impact:** Low-throughput uniaxial testing replaced w/ high-throughput imaging

Logistics

Teaching team

Prof. Trask

Muhammad Abdullah MEAM

Michelle Lin
MEAM

Alfredo Vasquez
BIOE

+ postdocs from my research group

+ collaborators visiting

How do you contact me?

Mainly: In class, in office hours, via Ed

Everything goes here

Email: ntrask@seas.upenn.edu

Please email me only for very private class matters or for non-class-related topics

Office Hours: TuTh 11-12

This week only me. Will post full OH schedule with TAs Friday

Course Mechanics

Weekly assignments on computing

1 in-class quiz during the semester (10/25)

1 open-ended final project/competition

We'll discuss ~mid-semester

Syllabus and Schedule posted on Github

ENGR 1050: Introduction to Scientific Computation
Fall 2023

(last updated: August 30, 2023)

Description

ENGR 1050 provides an introduction to computation and data analysis using Python – an industry standard programming environment. The course covers the fundamental of computing including variables, control structures, and functions. These concepts are illustrated through examples and assignments that show how computing is applied to various scientific and engineering problems. Examples are drawn from the simulation of physical and chemical systems, the analysis of experimental data, the simulation of dynamic systems, and control of sensors and actuators.

Course Objectives

By the end of this course, you should be able to:

- Translate an English simulation or design problem into a computational model
- Choose a numerical method for analyzing or simulating that engineering system
- Code and debug your chosen computational approach
- Produce a visualization for interpreting the results

Prerequisites

The course does not assume any prior programming experience but will make use of basic concepts from calculus and engineering. Relevant mathematical and engineering principles will be communicated in detail as needed. If you have doubts about your preparedness for the course, please visit the professor's OH.

Teaching Staff

Instructor:	Cynthia Sung Assistant Professor MEAM	crsung@seas.upenn.edu Office hours: M 3:15pm-4:15pm, R 12pm-1pm
Teaching Assistants:	Bibit Bianchini Jeremy Wang Niall Hosein Michelle Lin Tobia Ruth	Alice Li Josh Leshinskie Alexander Qi Alfredo Vazquez

Office Hours

The professor and TAs will hold office hours every week. We will announce these hours during the first week of class and also post them on our Google calendar and Ed Discussion. Please check the calendar for any updates throughout the semester.

ENGR 1050: Introduction to Scientific Computation
Course Schedule (Fall 2023)
(last updated: August 30, 2023)

Week	Dates	Topics	Assignments
1	8/28-9/3	Course introduction and mechanics, python intro, variables and variable types, basic math	
2	9/4-9/10	Plotting, lists/arrays	HW 1
3	9/11-9/17	Decision trees, conditionals, Boolean expressions, pseudocode	HW 2
4	9/18-9/24	Filtering, for loops, range, list comprehension	HW 3
5	9/25-10/1	Search, while loops, break, continue	HW 4
6	10/2-10/8	Modularizing code with functions, variable scope, interactive plots	HW 5
7	10/9-10/15	Numpy arrays and math	HW 6
8	10/16-10/22	OpenCV, image processing, masks, edge detection	HW 7
9	10/23-10/29	Video processing and tracking	Quiz 10/25
10	10/30-11/5	Scipy, Numerical integration, solve_ivp	HW 8
11	11/6-11/12	Polynomial fits	HW 9
12	11/13-11/19	Animations	HW 10 Final project proposal due
13	11/20-11/26	Putting it together	HW 11
14	11/27-12/3	Python scripts	HW 12
15	12/4-12/10	Comparison with MATLAB, MATLAB basics	
Finals	12/11-12/21	Final projects	Final projects

Class Format

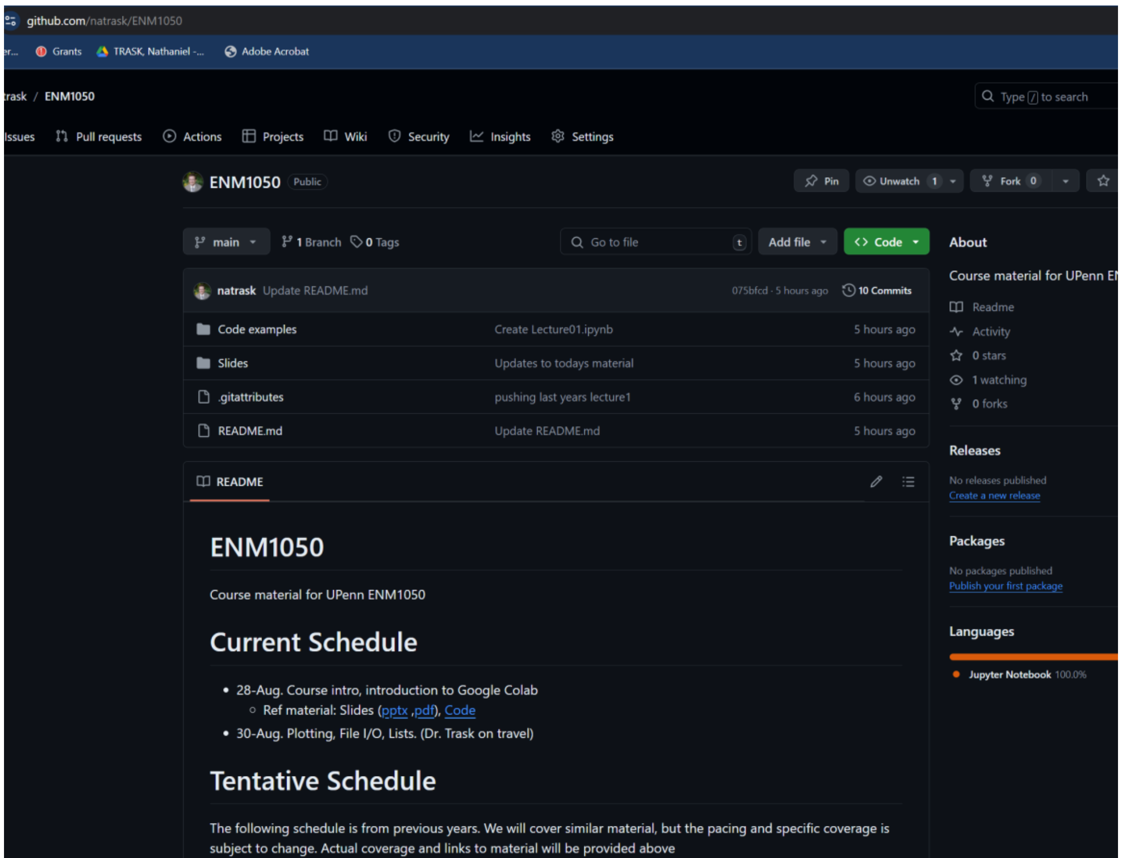
- This course covers scientific computation
- But it is also the first time many of you have programmed before

The most important thing is **PRACTICE**

- Lectures will start with a short presentation of the day's material, but the focus will be practice exercises
- Weekly homework assignments for you to practice coding longer programs on your own

How will we communicate?

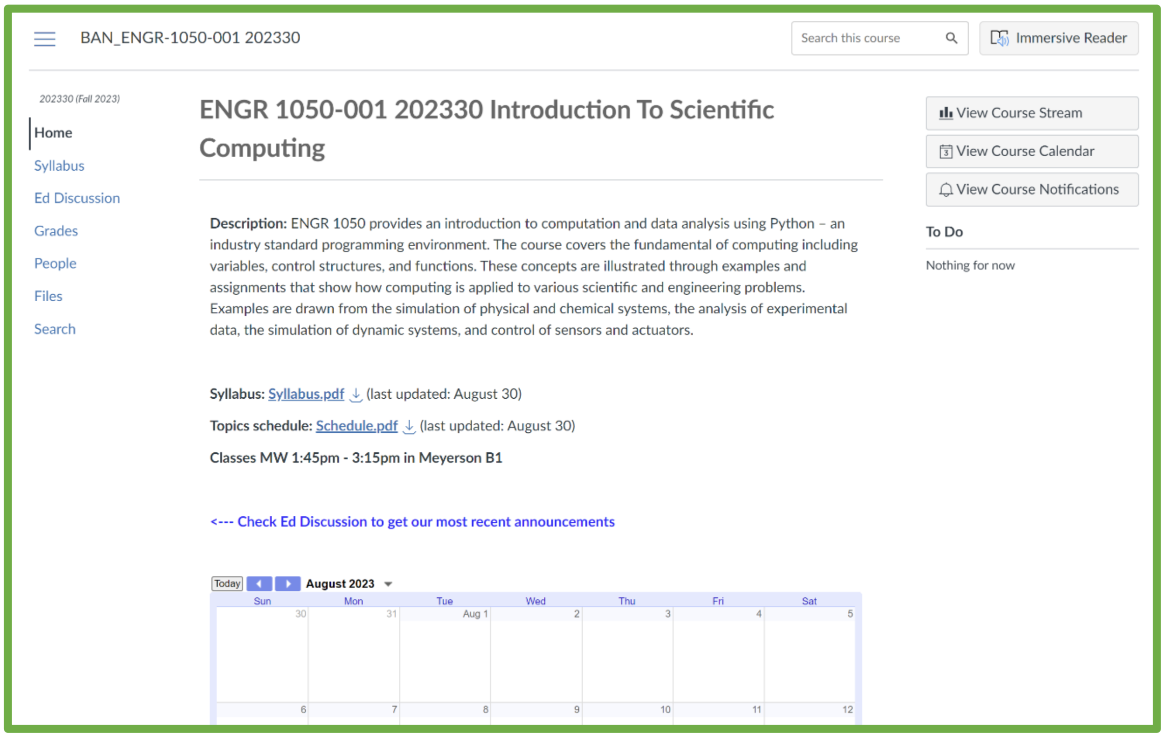
https://github.com/natrask/ENM1050



- All notes and material for the class will be linked through here
- One stop shop

How will we communicate?

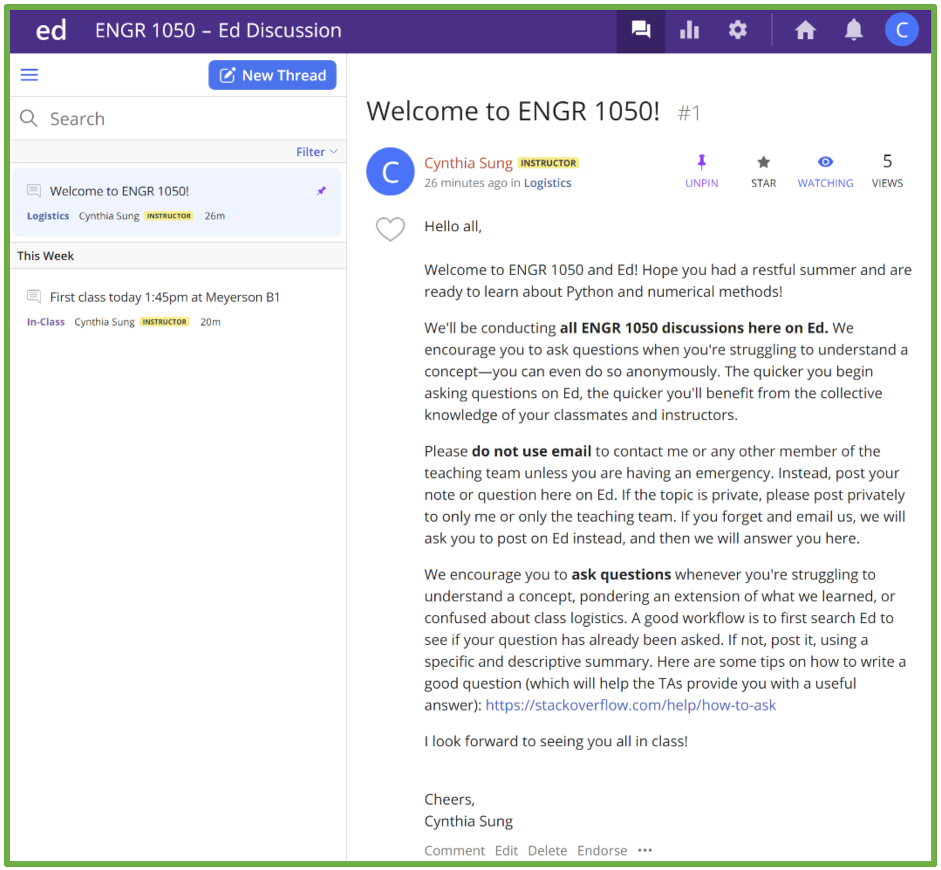
Canvas: https://canvas.upenn.edu



- Assignments submitted through here, linked by github
- Gradebook for assignments

How will we communicate?

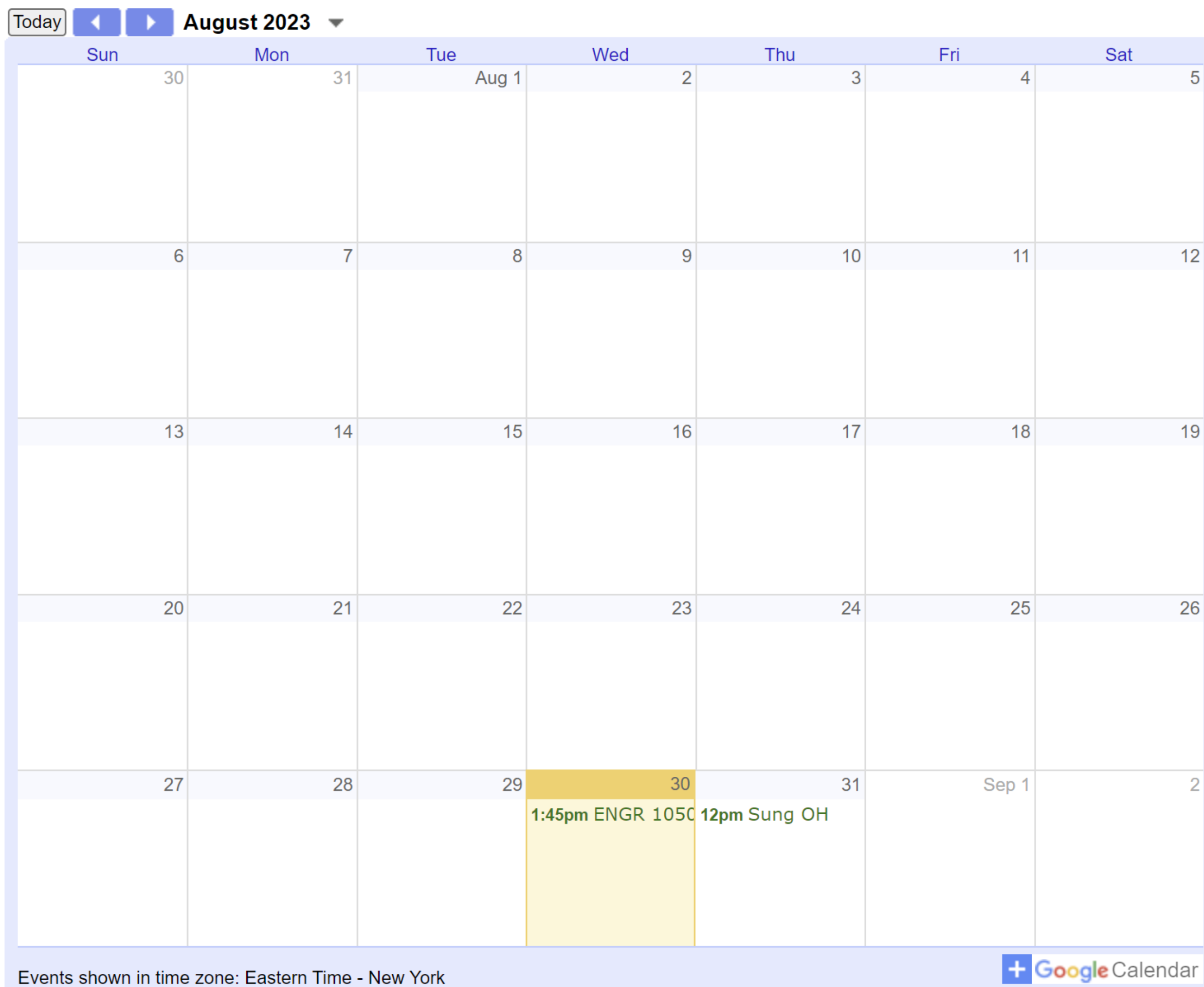
Ed Discussion



- See link in Github (not live yet)
- Questions and comments
- Post anonymously (hidden name) if you don't want other students to identify you
- Post privately (only to instructors) if you are asking about something personal

How will we communicate?

Google Calendar – office hours (once TA schedule set)



Collaboration Policy

I encourage you to work together.

Any work you submit should be your own work.

Acknowledge your collaborators.

Ask yourself how the collaboration enhanced your learning. If you have trouble answering this question, you are probably engaging in the wrong type of collaboration.

AI Collaboration Policy

I encourage you to use the tools you have available to you to learn the material.

You should understand any work that you submit.

Acknowledge your collaborators.

Realize that computers have their own limitations and use these tools thoughtfully.

Getting started in Python

What is a computer program?

A **computer program** is a **sequence of instructions** telling the computer how to perform a computation

A program:

1. Takes input from a user or file
2. Processes it
3. Produces output

Most importantly, **programming** is about **rules**

Computers do **exactly** what we tell them to do

What is a programming language?

A **programming language** is a **formal language** to express instructions.

Natural languages

- Syntax less crucial
- Ambiguous
- Redundant
- Idioms and metaphors

Programming languages

- Strict syntax
- No ambiguity
- Concise
- Literal

Writing a program

1. Identify problem
2. Make solution strategy
3. Break up solution into subtasks
4. Write down sequence of operations using syntax of the programming language
5. Test

Our tool of choice: Python

Python is a **high-level, general-purpose language** with a focus on **code readability**.

“Python is powerful... and fast; plays well with others; runs everywhere; is friendly & easy to learn; is Open.” – Python website

We start by using Google’s Colaboratory environment.

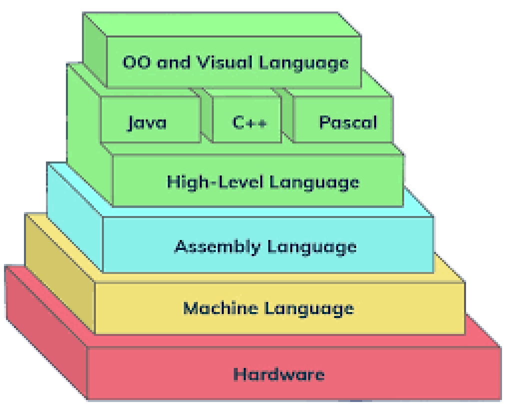
Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

– Colab website

What else is out there beyond python?

Fast, hard to program — Slow, easy to program



Non-scientific computing

What questions do you have?

Let’s get set up

In-class exercises

In-class exercises will normally be done in pairs.

For today only, I would like you to do them individually to make sure everyone learns how to interact with Colab.

You can work with someone if you were not able to bring your own computer.

Whatever you don’t finish in class today, you should finish at home and turn in the following Wednesday.

These are graded on a combination of completion and correctness.

We can `print` text

parentheses

function name

text in SINGLE QUOTES

```
print('this is a story about nat. he was
born and raised in MA. after finishing
high school, he moved to New Mexico. For
fun, he enjoys music and baseball, and he
studies scientific machine learning.')
```

What happens if I want to make the story about someone else?

Variables

Variables let us **refer** to things in our code.

Use "=" to **assign** values to variables

```
<variable name> = <variable value>
```

the **variable name** must appear on the **left**
the **value** must appear on the **right**

Displaying the story with variables

```
name = ‘nat'
pronoun = 'he'
```

use commas or + to combine text together

```
print('this is a story about', name+'. ',
pronoun, 'was born and raised in MA.
after finishing high school,', pronoun,
'moved to New Mexico. For fun, ‘,pronoun,
'enjoys music and baseball, and ',
pronoun, 'studies scientific machine
learning.')
```

Add as many variables as you need to make your program work the way you want it to

Variable names

Must

- start with a letter or underscore
- contain only letters, numbers, or underscores
- not be keywords (e.g. `print`)

Are case sensitive

- `myVariable` ≠ `myvariable`

Which are valid variable names?

RULES
- start with a letter or underscore
- contain only letters, numbers, or underscores
- not be keywords

```
and         my-name

_and        your_name

var         COLOR

var1

1var
```

Basic types in Python

`str` (string)

- Ex: “`ENGR 1050`", 'a’, “`nat`“

`int` (integer)

- Ex: `2`, `73`, `-122`

`float` (floating point number)

- Ex: `3.14`, `-18.0`

`bool` (boolean)

- can be `True` or `False` only

Arithmetic operators

Applied to numbers (int, float), produces numbers

+	addition
−	subtraction
*	multiplication
/	division
**	exponentiation
%	modulo (remainder)

Relational operators

For comparing two variables

Applied to multiple types, always produces a `bool`

`x == y` `True` if `x` equals `y`

Don't confuse with assignment operator (`x = y`)!

`x != y` `True` if `x` does not equal `y`

`x > y` `True` if `x` greater than `y`

`x < y` `True` if `x` less than `y`

`x >= y` `True` if `x` greater than or equal to `y`

`x <= y` `True` if `x` less than or equal to `y`

Boolean (logical) operators

Applied to booleans, produces booleans

`x and y`	`True` if both `x` and `y` are `True`
`x or y`	`True` if either `x` or `y` are `True`
`not x`	`True` if `x` is `False`

What does this code output?

```
a = 19
b = 21
a_plus_b = a + b
res_div_5 = a_plus_b / 5
rem_div_2 = res_div_5 % 2
is_even = (rem_div_2 == 0)
print(is_even)
```

What about `5/2`?

Python performs floor division

- If both numbers are integers, the result is an integer
- Floor division chops off the fractional part of the result

To get a fractional result

```
5.0/2
float(5)/2
```

Let’s write something together

Two trains are on the same track, approaching each other at a speed of 50 mph and 80 mph, respectively. They are currently 300 miles apart. How long before they collide?

Some notes on Approach and Style

Debugging

Programming is complicated! We all make mistakes.

Errors are called **bugs**.

Finding and **removing bugs** is called **debugging**.

Code style

To limit errors and make debugging easy, your code should always be

- readable
- consistent
- commented and/or documented

Code that is easy to read and understand is far more likely to be correct!

Code approach

- Start with a plan
- Outline your code using comments (pseudocode)
- Fill in chunks of the code and check as you go

Style: Comments

Use # to add notes for yourself and others

2 types:

- Headers – go at the top of your code and summarize the function of the whole program
- In-line – go inside your code and explain particular lines/chunks

Rule of thumb: If you come back in 1 year, what will need to be explained?

Rule of thumb 2: ~1 comment every 5-10 lines.

Style: Variable names

Should indicate purpose (be descriptive)

```
a = 3            # what the heck is a??
num_apples = 3   # oh, the number of apples
```

It's okay to use standard variable names *if you comment*

```
v = 5            # velocity in meters/sec
vel = 5          # velocity in meters/sec
```

Should follow naming conventions

(we use underscore_case)

- start with a lower case letter
- start each "new word" with an underscore

```
my_variable = 7
my_variable_with_a_very_long_name = 6.177
```

Style: Operators

Use spacing and parentheses to improve readability

```
answer = num2*7+num1/6-27*num2

answer = (num2 * 7) + (num1 / 6) - (27 * num2)
```

Order of operations is easy to forget. Use parentheses!

My story

```
print("this is a story about", \
    name, \
    ".", \
    pronoun, \
    “was born in MA in 1985. when", \
    pronoun, \
    “graduated high school,", \
    pronoun, \
    "moved to New Mexico, where", \
    [...])
```

Summary

In class, we

- Created some variables
- Manipulated some variables
- Looked at variable types
- Did some math

Exercise (this will be HW0):

Rewrite Nat’s story to tell your own by defining variables to swap out your own details