Module 01
Course Syllabus, Prerequisites, Applications, Course Overview

Ahmad Taha — Spring 2026

CE 2989 — Numerical Methods in Civil & Env. Engineering

Email: ahmad.taha@vanderbilt.edu

Webpage: http://lab.vanderbilt.edu/taha

VANDERBILT UNIVERSITY

January 6, 2026

Course Instructor: Background & Interests

Background

- Born and raised in Beirut, Lebanon
- Finished my Ph.D. in ECE from Purdue University in August 2015
- Undergraduate education: American University of Beirut — Class of 2011, B.E., ECE
- Assistant Professor, ECE Department @ UTSA, August, 2015—August 2021
- Associate Professor, CEE Department @ Vandy, August 2021—currently

What do professors do and my lab's objective

Algorithms for better infrastructure

Examples on my group's recent research

- Energy systems, climate change, and transitioning to fossil fuels-free systems
- Everything water: distribution, drinking, stormwater, flood control, etc...
- Traffic control and monitoring, electrifying transportation
- Cyber-security of infrastructure
- Some theoretical questions
- Almost all of these topics require some sort of numerical computing, to be covered in this class

Module 01 Outline

1. Course syllabus and expectations
2. Homework 1
3. Course outline
4. The fun stuff starts — we will introduce numerical methods and define the class scope

Part I — Course Syllabus and Outline

Course webpage & Communication

Course Page:

- Vanderbilt Brightspace: https://brightspace.vanderbilt.edu
- *Email is the best form of communication!*

Office Hours:

- Tuesdays and Thursdays, 12:30PM–02:00PM
- Or by appointment
- Location: 293 Jacobs Hall

TA Info:

- Gavin Blair
- Office hours: MWFs: 11AM–1PM
- Location: TBA + Zoom Link https://vanderbilt.zoom.us/j/4465422480

Course Description

Introduction to concepts, algorithms, and methods of numerical analysis, including: *(i)* linear algebra and matrix computations concepts; *(ii)* solving linear and nonlinear systems of equations; *(iii)* matrix decompositions; *(iv)* Newton's method and its variants; *(v)* stability and convergence of numerical methods; *(vi)* numerical interpolation and integration; *(vii)* finite-difference methods for solving differential equations; *(viii)* optimization problems and linear programming. Applications to basic problems in civil and environmental engineering.

Main References

I will be producing lecture notes, guided by these two textbooks

1. Chapra, Steven C. and Canale, Raymond P., *Numerical Methods for Engineers, 8th edition.* Mcgraw-hill, 2021.
2. Chapra, Steven C., *Applied numerical methods with MATLAB for engineers and scientists*. Mcgraw-hill, 2018.

and some online references that I will be reproducing and providing to the students. The two textbooks are very similar and are available as e-books online. I will be using figures from the book (FYI, so you don't think I made all figures)

Course Objectives & Expected Outcomes

The course presents an introduction to numerical computing methods in engineering. This topic is needed for any engineer, so it is important to cover the fundamentals. The class basically integrates five topics in one: algorithms, programming, linear algebra, differential equations, and engineering applications. The course objectives are as follows:

- Introduce students to classical numerical methods available for problem solving with applications in civil and environmental engineering.
- Introduce students to concepts such as precision, errors, and tolerances in algorithms as well as convergence analysis and its impact on solving numerical computational problems.
- Enhance students' knowledge of linear algebra, calculus, and differential equations.
- Improve the understanding of algorithms and processes.
- Improve basic programming skills and the coding of functions that solve numerical problems.
- Introduce students to mainstream algorithms that perform solving systems of linear equations and numerical integration.
- Introduce students to the concept of mathematical optimization of convex problems.
- Cover applications of optimization and computing in CEE.

Prerequisites

An undergraduate-level understanding of:

- Some basic programming (CS 1100 or 1101 or 1103).
- Corequisite: MATH 2420, although maybe not so much needed to be honest.
- Do well on this class, and your life will be easier later on.

Grading Policy

- Homework assignments (0%) ????
- Midterm 1 (25%): Wednesday February 11, 2026, at 6pm.
- Midterm 2 (25%): Wednesday March 25, 2026, at 6pm.
- Final Exam (40%): Thursday April 23, 2026, at 2pm.
- Attendance and instructor evaluation (10%)

Why this grading policy?

Course Grade Cutoffs

- A−, A, A+: 85–100
- B−, B, B+: 70–84
- C−, C, C+: 55–69
- D−, D, D+: 40–54
- F: ≤ 39

Extra Sessions

- We will have **biweekly extra sessions** to supplement the main class lectures
- **Timing**: few days before the *due date of the homework*
- **Purpose**: going through any homework questions you might have
- + Reviewing basic concepts from the lecture
- + Review for the midterm/final
- **First session**: January 12, 4:30pm, via Zoom with Gavin https://vanderbilt.zoom.us/j/4465422480

Programming Tools

- MATLAB will be required for homework assignments and course projects
- Students can obtain the discounted student version of MATLAB
- All homework assignments should ideally be typed. It's encouraged to use LaTeX for homework assignments (honestly, there's no good reason not to!).

Class Policies

- Regular attendance
- Emailing me
- Showing up early (or not showing up at all)
- Smartphone breaks
- Laptop usage in class
- On the homework
- Changes to the syllabus

Tentative Course Schedule

Module I — Class Overview & Background ..... ≈ 1 class
- Course introduction & syllabus, topics to be covered, and applications

Module II — Mathematical Modeling, Matlab, and Error Analysis ..... ≈ 3–4 classes
- Revision of basic mathematical concepts and introduction of error analysis

Module III — Finding Roots ..... ≈ 4–5 classes
- Computing roots of equations $f(x) = 0$ via bisection, Newton, and other methods

Module IV — Solving linear system of equations ..... ≈ 3–4 classes
- Methods to solve $Ax = b$ and matrix factorization

Module V — Curve Fitting, Interpolation, and Fourier Analysis ..... ≈ 3–4 classes
- Where we learn some fundamental methods used in the Anthropocene's digital age

Module VI — Numerical Integration and Differentiation ..... ≈ 4–5 classes
- Because like who wants to integrate manually?!

Module VII — Numerical Methods to Solve ODEs ..... ≈ 3–4 classes
- You see how computers solve ODEs (spoiler alert: Matlab didn't take the ODE class)

Module VIII — Numerical Methods to Solve PDEs ..... ≈ 3–4 classes
- Not sure we will have time for this but let's at least try

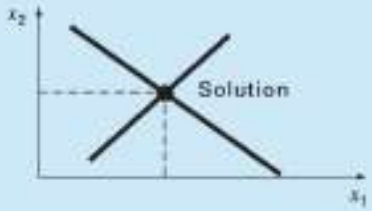
Course Outline

(a) *Part 2:* Roots of equations
Solve $f(x) = 0$ for x.

(b) *Part 3:* Linear algebraic equations
Given the a's and the c's, solve

$$a_{11}x_1 + a_{12}x_2 = c_1$$
$$a_{21}x_1 + a_{22}x_2 = c_2$$

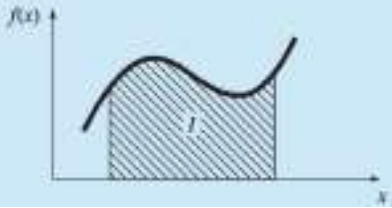
for the x's.



(c) *Part 4:* Optimization
Determine x that gives optimum $f(x)$.

(d) *Part 5:* Curve fitting

(e) *Part 6:* Integration
$I = \int_a^b f(x)\,dx$
Find the area under the curve.



(f) *Part 7:* Ordinary differential equations
Given

$$\frac{dy}{dt} = \frac{\Delta y}{\Delta t} = f(t, y)$$

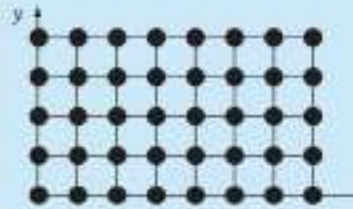
solve for y as a function of t.

$$y_{i+1} = y_i + f(t_i, y_i)\,\Delta t$$

(g) *Part 8:* Partial differential equations
Given

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

solve for u as a function of x and y



Homework #1

- Learn how to produce nice math-heavy documents
- Review programming fundamentals on Matlab, read Chapters 1–3 in Chapra-Matlab
- Difficult to be a great engineer without knowing basic programming
- No fancy programming in this class—just calling basic functions and writing high-level algorithms in simple functions on Matlab
- Homework is uploaded (theoretically)
- In the age of LLMs and AI, no excuse for not knowing how to program: LLMs can fix your buggy code, so utilize it
- How LLMs helped my student reproduce a year's worth of work in 2 hours

The History

- It's a bit nice to see how the relationship of humans and computing has evolved
- History of computing can be divided into four eras
  1. Pre-mechanical era
  2. Mechanical era
  3. Electro-mechanical era
  4. Electronic digital era

Pre-Mechanical Era (Prehistoric Era)

- Prehistoric era defined prior to humans developing writing methods
- Prehistoric age: Stone, Bronze, and Iron Ages
- Computing tools:
  - From counting on fingers
  - To hash marks in sand, then pebbles, hash marks on walls, and hash marks on bone
- Historic evidence: artifacts like the *Ishango bone* and *clay tokens* reveal early mathematical thinking

Mechanical Era

- From the days of the Abacus (4000 BCE)

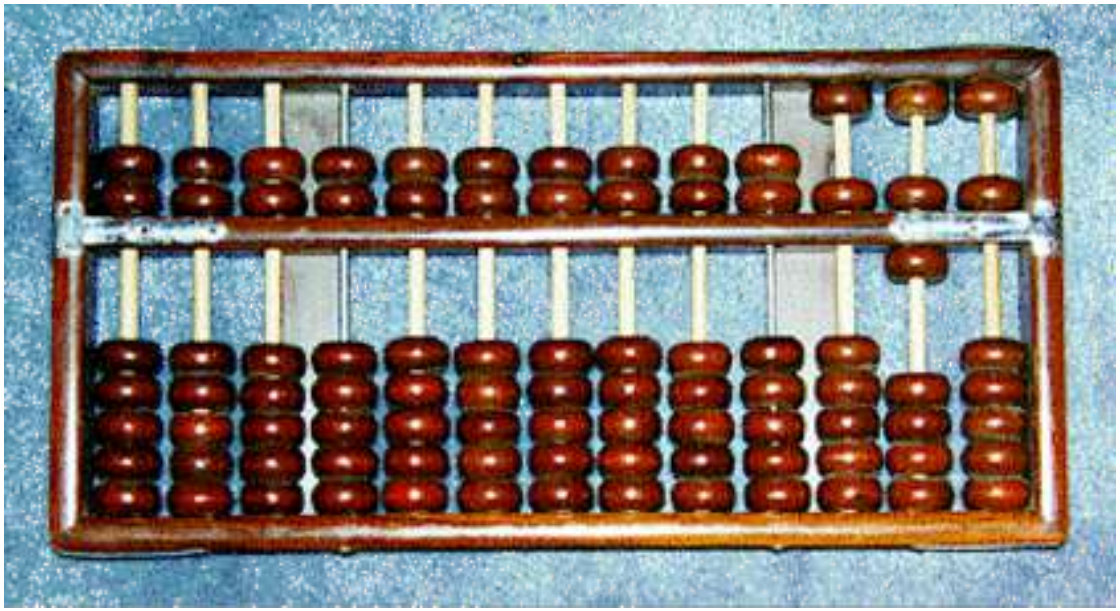


Figure: The Abacus

- To Charles Babbage's Difference Engine (1812 CE)
- So many examples and mechanical apparatuses that vary in complexity, purpose, sophistication
- Humans needed to do more computing as we progressed

Electro-Mechanical Era

- Basically this era initiated the idea of computers
- From Herman Hollerith's 1890 *Census Counting Machine*

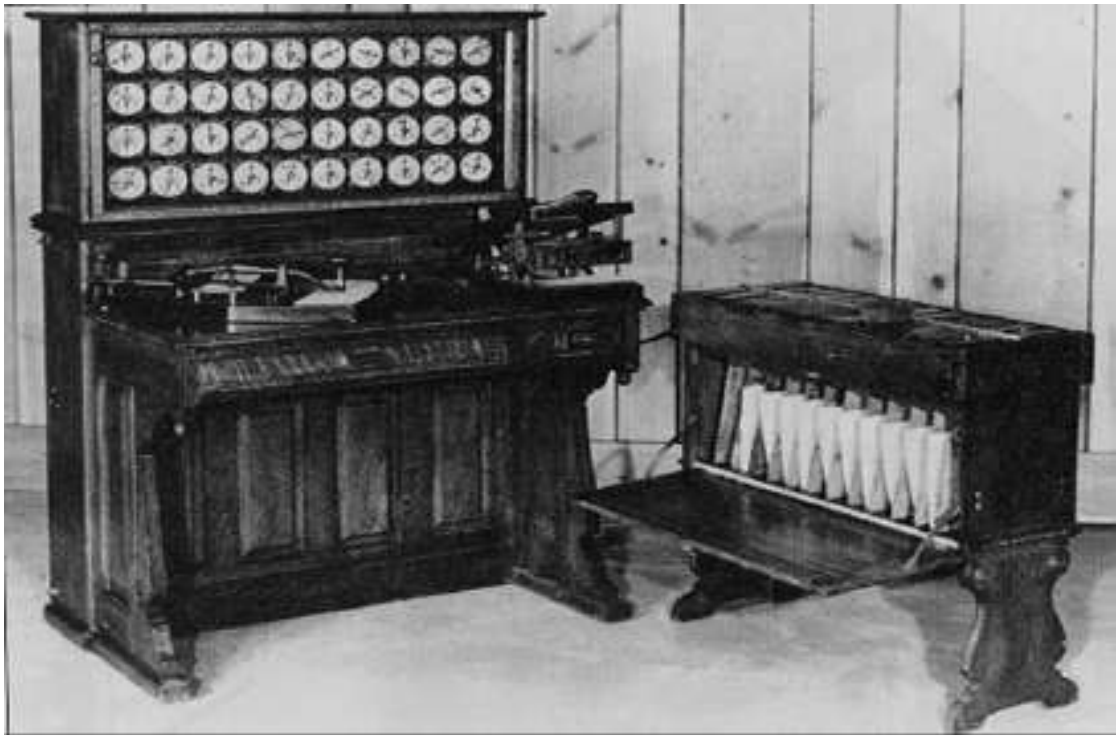


Figure: Census Counting Machine

- To Howard Aiken and *Harvard Mark I* [also knowns as IBM Automatic Sequence Controlled Calculator (ASCC)] (1944)

IBM's ASCC

Figure: IBM's Automatic Sequence Controlled Calculator

- This machine weighed 5 tons and had 500 miles of wiring

Electronic Digital Era

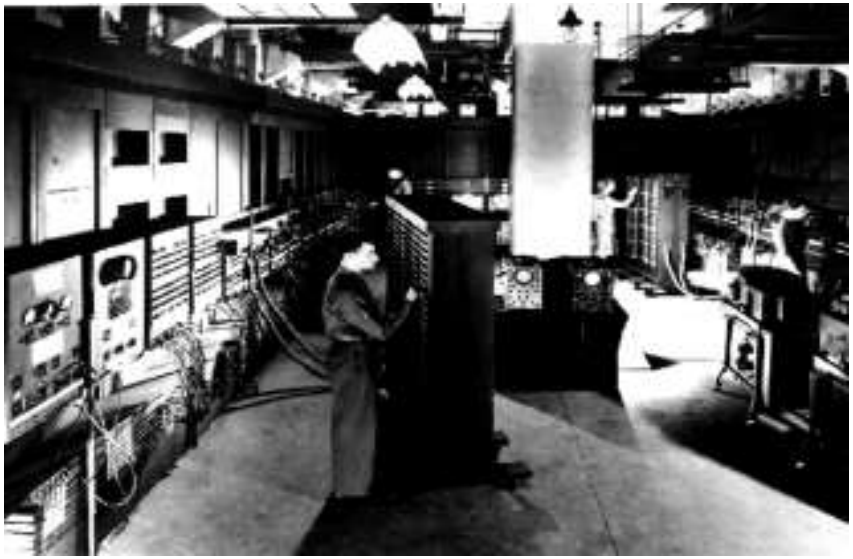


Figure: The ENIAC: Electronic Numerical Integrator and Computer

- ENIAC: 30 tons, 18,000 vacuum tubes...computing power of a calculator
- Programmed by rewiring the machine with detailed instructions
- Lady programmers were known as “computers”, gendered times

Moore's Law and Today's Computing

Figure: Moore's law: compute power (or number of transistors in an integrated circuit) doubles every two years?

How did we do so?? So many innovations

- Moore's law does *not* mean that transistors (or circuit elements) get twice as small in area every two years
- The sad reality of transistors oligopoly...how does this relate to computing?

Progress in Chip Manufacturing

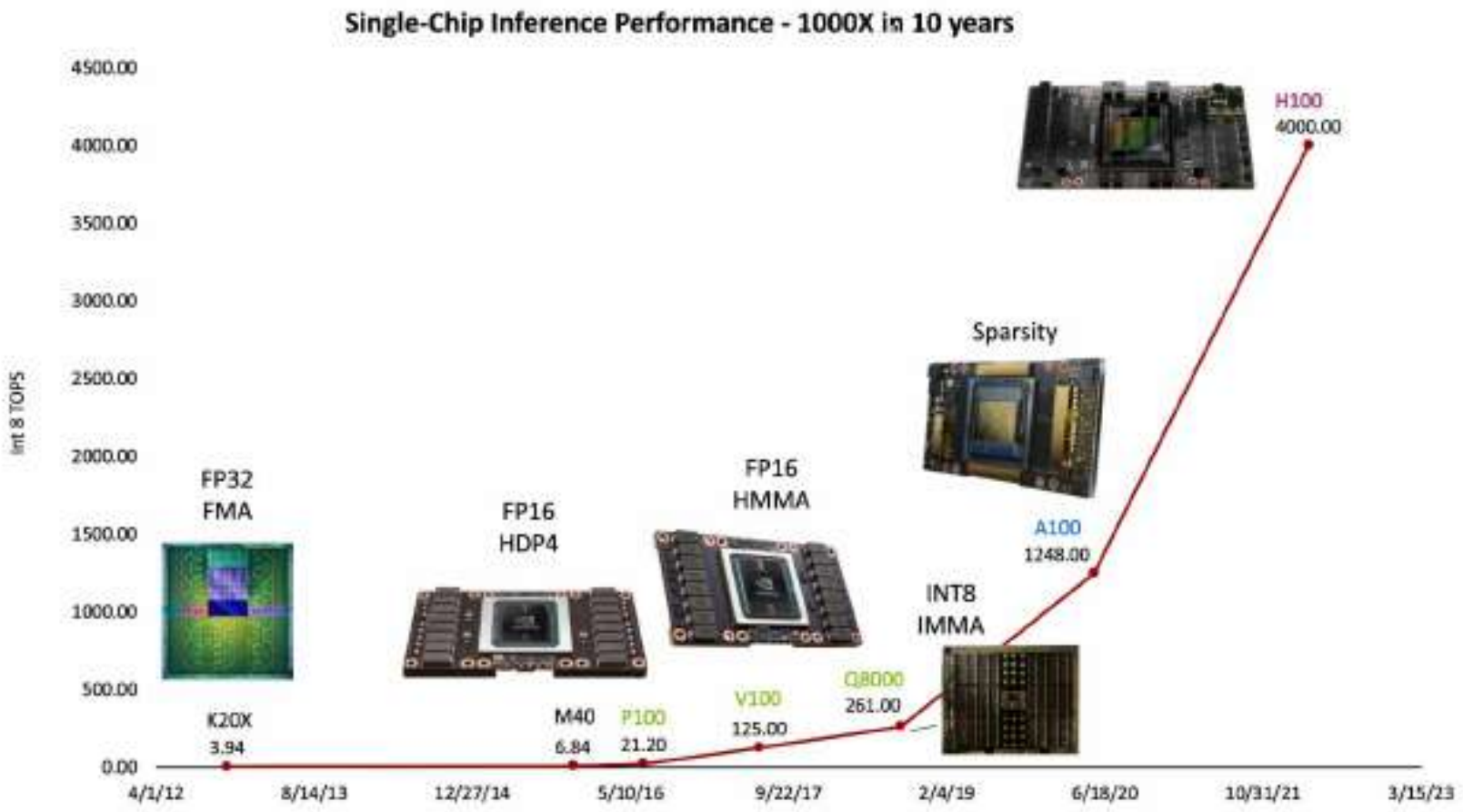


Figure: Crazy improvements in compute power.

The Stock Market and Your Parents' 401K

- What is the AI revolution in reality? It's a computing revolution
- More compute power $\Rightarrow$ better AI + better prediction
- What is AI anyway?????
- It is a fast way to do inference using fast numerical computing algorithms
- The bet on computing and AI: how 5–10 companies dominate the market
- How investment portfolios of your parents are no longer diversified
- The risky business of AI-betting

Definition

- Computing is defined as: STUDY OF SYSTEMATIC PROCESSES THAT DESCRIBE AND TRANSFORM INFORMATION: THEIR THEORY, ANALYSIS, DESIGN, EFFICIENCY, IMPLEMENTATION, AND APPLICATION.
- Fundamental question: WHAT CAN AND CANNOT BE AUTOMATED?[1]
- This class is about automation: automating algorithms that solve specific math problems that are relevant in various engineering applications

[1] Denning et. al., *Computing as a Discipline*, Communications of the ACM, January, 1989

Why We Need Computing?

- Classically, scientists and engineers obtained analytical solutions for problems
- Closed form solutions, provably accurate
- Cannot do that with large-scale systems, infrastructure, very complex problems
- This motivates automated numerical computing

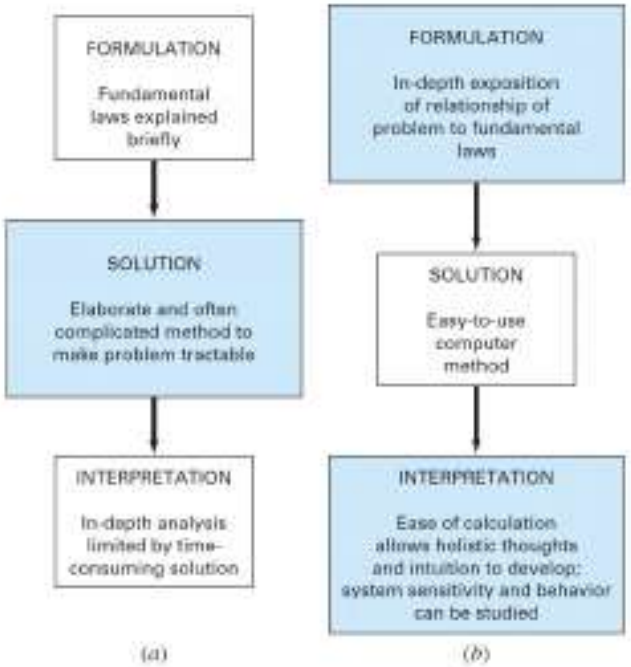


Figure: Old versus new: how computing has changed.

Models, We Need Models

- To help with various computing problems, we need a model
- Not like supermodels, but math/physics/chemistry/biology models
- Models help us learn the world, predict, optimize, design, etc..
- But models can be used to distract and deceive: models are morally neutral so remember that
- Models of engineering systems versus models of nature
- Computing and engineering problem solving requires
  - Understanding of engineering systems (model)
  - Observation + experiment
  - Theoretical analysis + generalization + testing
- Computers are great to automate computing, but they require user input (model, data, analysis)

Mathematical Models

- Math model can be represented via

$$\begin{matrix}\text{Dependent}\\ \text{Variable}\end{matrix} = f\left(\begin{matrix}\text{independent}\\ \text{variables,}\end{matrix} \quad \text{parameters,} \quad \begin{matrix}\text{forcing}\\ \text{functions}\end{matrix}\right)$$

  or

$$y = f(t, x, u)$$

- **Dependent variables**: Characteristic that usually reflects system state
- **Independent variables**: Dimensions such as time/space
- **States**: reflect the system's properties/states
- **Inputs**: external influences acting upon the system
- **Example:** Newton's second law [sic] of motion

$$\sum_i F_i(t) = ma(t)$$

All Models Are Wrong; Some Are Useful

- George Box's famous line
- Important to always remember that a model is a model which is a model
- Models vary in complexity: finite-element, PDEs, ODEs, input-output, black box, etc..
- Example (g is gravity, m is mass, c is drag coefficient):

$$\frac{dv}{dt} = \frac{F}{m}$$

$$F = F_D + F_U$$

$$F_D = mg$$

$$F_U = -cv$$

$$\frac{dv}{dt} = \frac{mg - cv}{m}$$

Parachutist Velocity Prediction

- Parachutist problem: predicting velocity, distance to landing
- The previous model predicts that, it's a simple one though
- This is a first order ordinary differential equation (ODE):

$$\dot{v}(t) = dv/dt = g - (c/m)v(t)$$

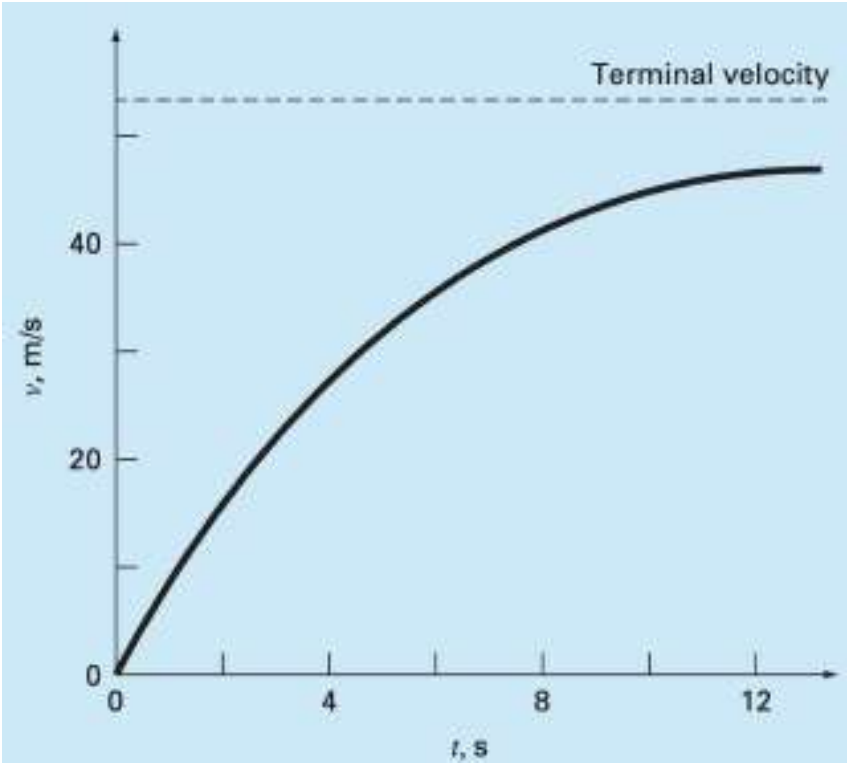
- Has a unique solution:

$$\boxed{v(t) = \frac{gm}{c}\left(1 - e^{-\frac{c}{m}t}\right)}$$

- This assumes zero starting velocity $v(t_0) = v(0) = 0$
- Solution in the box is called *analytical solution*, it's exact
- This is awesome, but almost all models in CEE problems DO NOT have analytical solutions (or you need a Math PhD to obtain one or the most expensive LLM/AI assistant)
- This class: teaches you how to solve problems when exact solutions are impossible/hard/intractable to find
- Relevance to CEE and systems-level problems

Example

- Consider a person with $m = 68.1$kg jumping from a stationary hot air balloon
- Assume drag coefficient $c = 12.5$kg/s
- This results in:

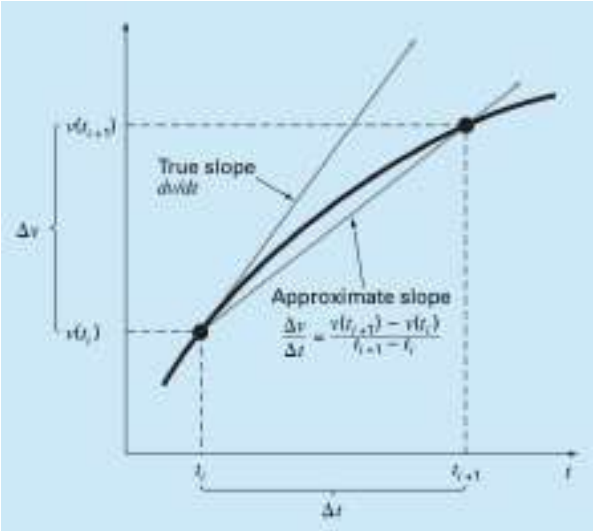


Can We Approximate Solutions?

- This class will teach you how to find approximate solutions
- One way: approximate the derivative as follows:

$$\dot{v}(t) = dv/dt = \lim_{\Delta t \to 0} \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i = \Delta t} \underbrace{\approx \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\text{if } \Delta t \text{ is small enough}} \approx g - (c/m)v(t_i)$$

- This method is called Euler's method, forward Euler, finite difference, etc..
- Only works nicely if the discretization time-step difference is small



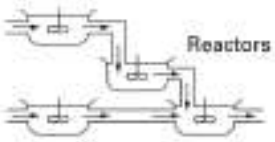
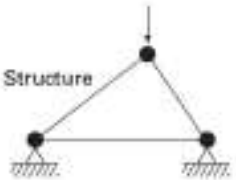
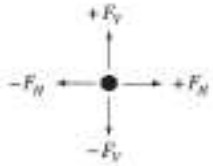
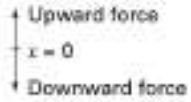
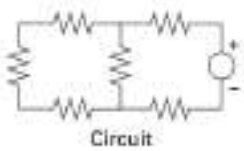
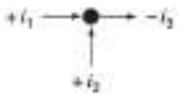
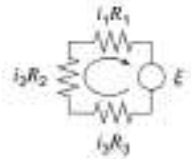
Closed-form Approximation: Euler's Method

- The approximate solution can be written as:

$$v(t_{i+1}) = v(t_i) + \underbrace{(t_{i+1} - t_i)}_{\Delta t}(g - \frac{c}{m}v(t_i)) = \alpha v(t_i) + \beta$$

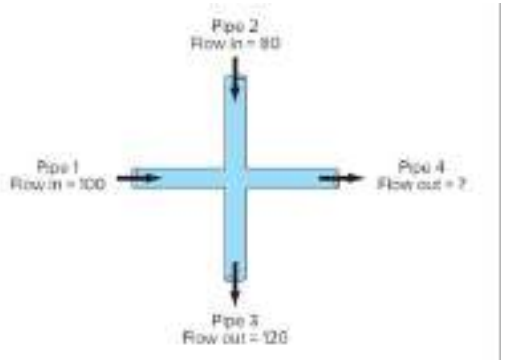
- Δt is called step-size, time-step, discretization parameter, sampling time, etc..
- Basically: new-value $=$ old-value $+$ slope x step-size
- Should you choose a small or large step-size Δt?
- Why not choose super super small Δt??
- Btw, this model is linear...what if the model is more realistic and nonlinear?

Models in Engineering

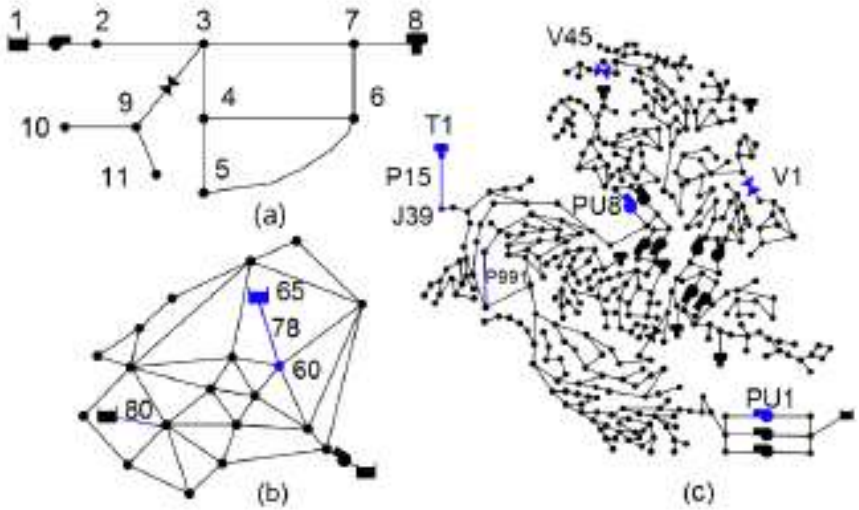
Field	Device	Organizing Principle	Mathematical Expression
Chemical engineering		Conservation of mass	Mass balance:  Over a unit of time period $\Delta\text{mass} = \text{inputs} - \text{outputs}$
Civil engineering		Conservation of momentum	Force balance:  At each node Σ horizontal forces $(F_H) = 0$ Σ vertical forces $(F_V) = 0$
Mechanical engineering		Conservation of momentum	Force balance:  $m\frac{d^2x}{dt^2}$ = downward force − upward force
Electrical engineering		Conservation of charge	Current balance:  For each node Σ current $(i) = 0$
		Conservation of energy	Voltage balance:  Around each loop Σ emf's − Σ voltage drops for resistors = 0 $\Sigma\,\xi - \Sigma\,iR = 0$

Focus on CEE Problems

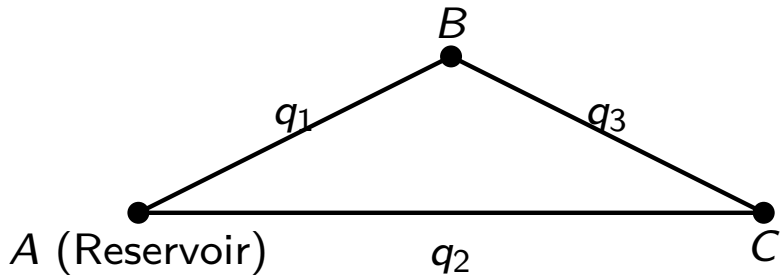
- We will focus in this class on applications in CEE



- Solving for flow in Pipe 4? Can write flow-balance equations
- Can we do it for a network of pipes?? Need to write $Ax = b$



Example from water systems



- Consider a simple water distribution network (WDN) with three pipes (with flows q_1, q_2, q_3):

$$A \leftrightarrow B, \quad A \leftrightarrow C, \quad B \leftrightarrow C.$$

  and two demand nodes

$$d_B = 0.01 \text{ m}^3/\text{s}, \qquad d_C = 0.012 \text{ m}^3/\text{s}.$$

- Also, assume Reservoir A has a fixed head/pressure $H_A = 100$ m
- What are the hydraulic equations governing water flow and pressure in this simple network?
- In water systems, two phenomena should be modeled: water mass (whatever comes in goes out) and energy conservation (energy loss/gain)
- Both are considered "physics-based" models
- Remember the physics-based versus nature-based models discussion

Example (Cont'd)

- **Solution:** For mass balance, the equations can be written as:
  – At nodes B and C:

$$q_1 - q_3 = d_B, \quad q_2 + q_3 = d_C$$

- For energy balance between two junctions i and j, we usually write $\Delta h_{ij} = h_i - h_j = R_{ij} q_{ij} |q_{ij}|$ where R_{ij} is pipe friction constant
- Then energy conservation around loop $A \to B \to C \to A$ gives:

$$h_{AB}(q_1) + h_{BC}(q_3) - h_{AC}(q_2) = 0.$$

- Which allows us to obtain the following nonlinear system of equations

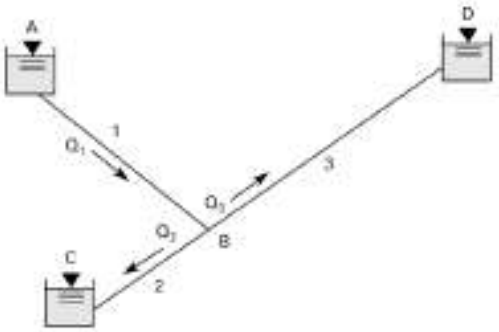
$$\begin{aligned} q_1 - q_3 - d_B &= 0, \\ q_2 + q_3 - d_C &= 0, \\ r_1 q_1 |q_1| + r_3 q_3 |q_3| - r_2 q_2 |q_2| &= 0. \end{aligned}$$

- How many equations and how many unknowns?
- How to solve the above nonlinear system of equations $f(x) = 0$?
- Linearization and Newton's method
- What happens if you have 1000s of junctions and pipes? How do you know you will get a solution?
- How to incorporate pumps and valves? Different types of pipes? Leaks?

A Question from Civil Engineering PE Exam

- The previous example may appear on your PE exam
- An example from an actual PE exam:

Three water reservoirs are connected by a branched piping system as shown, with piping data as tabulated.



pipe	L (ft)	D (ft)	f	C
1	2500	0.500	0.020	2
2	1600	0.333	0.025	3
3	3300	0.500	0.018	7

The elevations are 65 ft at reservoir A, 5 ft at reservoir C, and 43 ft at reservoir D. Friction factors are constant for each pipe. The volumetric flow rate in pipe 3 is most nearly

✗ A. 0.01 ft^3/sec

You did not select. This answer is incorrect.

✗ B. 0.70 ft^3/sec

You did not select. This answer is incorrect.

◉ C. 0.28 ft^3/sec

✓ This answer is correct.

✗ D. 0.38 ft^3/sec

You did not select. This answer is incorrect.

Module Summary

- We learned what models are
- All models are wrong (including Newton's method)
- But a model with 20% error is better than nothing
- But sometimes your grandma's intuition is better than a model
- Not all models are created equal
- This class will teach you how to automate solutions of ODEs or integrations or systems of equations, that are generated from simple physics-based models (because that's the best we can do)
- Sometimes problems are *static* (no time-dimension) like water flow balance (find x s.t. $Ax = b$ or $f(x) = 0$)
- Other times problems will involve time and hence they're *dynamic* or *time-dependent*
- Next: getting ourselves ready with Matlab and formulating error analysis