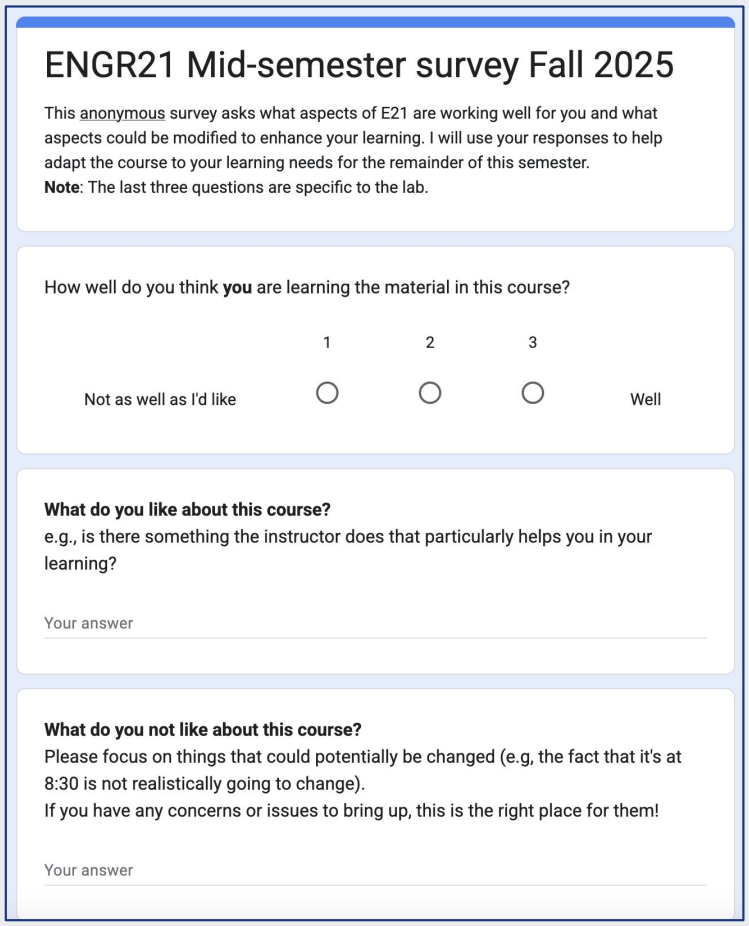
ENGR 21:
Computer Engineering Fundamentals

Lecture 11
Tuesday, October 07, 2025

Mid-Semester Survey



ENGR21 Mid-semester survey Fall 2025

This anonymous survey asks what aspects of E21 are working well for you and what aspects could be modified to enhance your learning. I will use your responses to help adapt the course to your learning needs for the remainder of this semester.
Note: The last three questions are specific to the lab.

How well do you think **you** are learning the material in this course?

	1	2	3	
Not as well as I'd like	○	○	○	Well

What do you like about this course?
e.g., is there something the instructor does that particularly helps you in your learning?

Your answer

What do you not like about this course?
Please focus on things that could potentially be changed (e.g, the fact that it's at 8:30 is not realistically going to change).
If you have any concerns or issues to bring up, this is the right place for them!

Your answer

Also available on class website

Numerical Methods for Engineering using Python

Numerical Methods

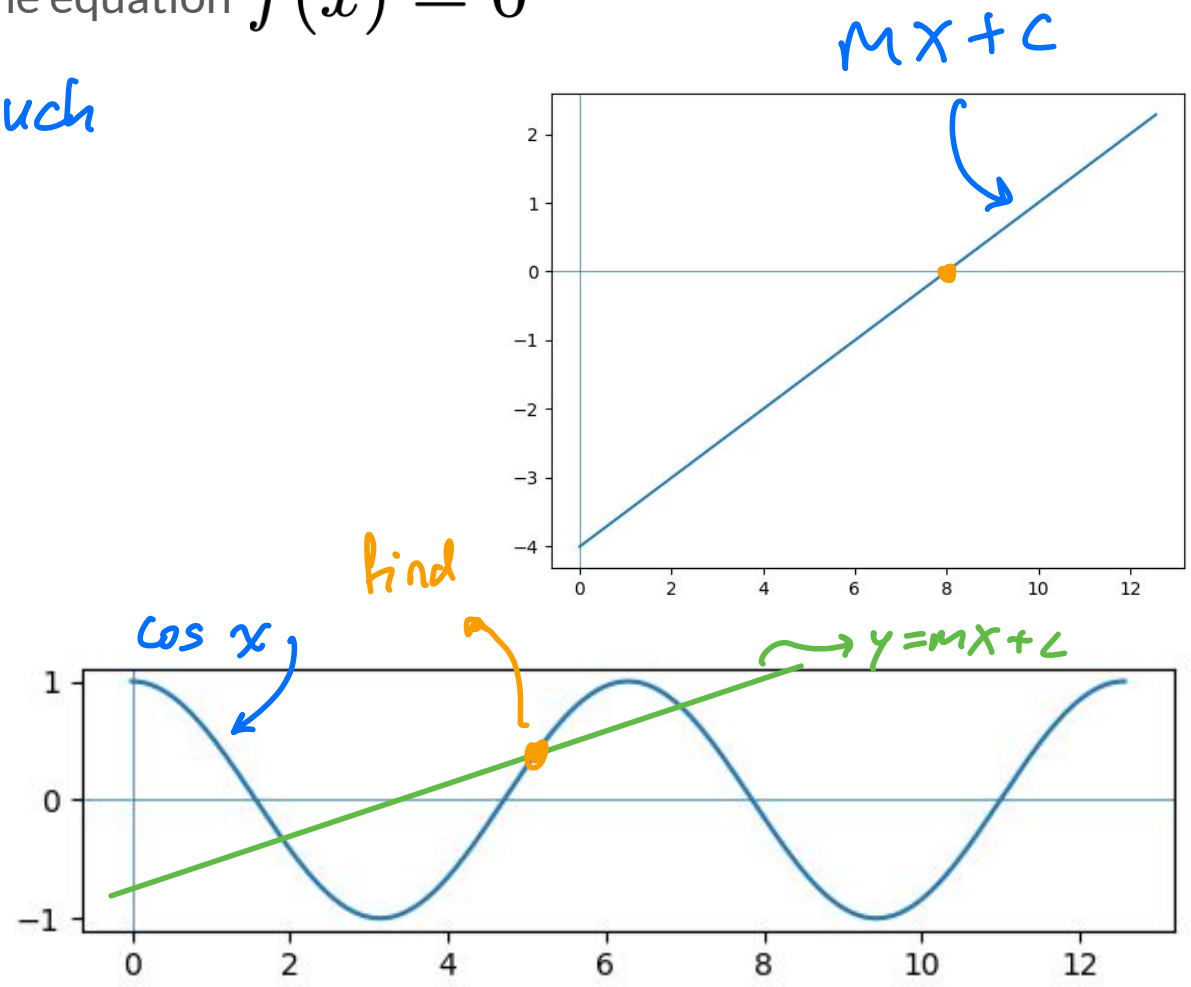
What exactly does that mean?

- Turn engineering problems into mathematical problems
- Turn mathematical problems into computational problems
- Teach a computer how to solve the computational problems
- Learn what **you** must do vs what the computer can do.
- Become:
  - Users of numerical methods
  - Developers of numerical methods

Root-Finding

For a given function f(x), solve the equation $f(x) = 0$

Find value of x, such that $f(x) = 0$

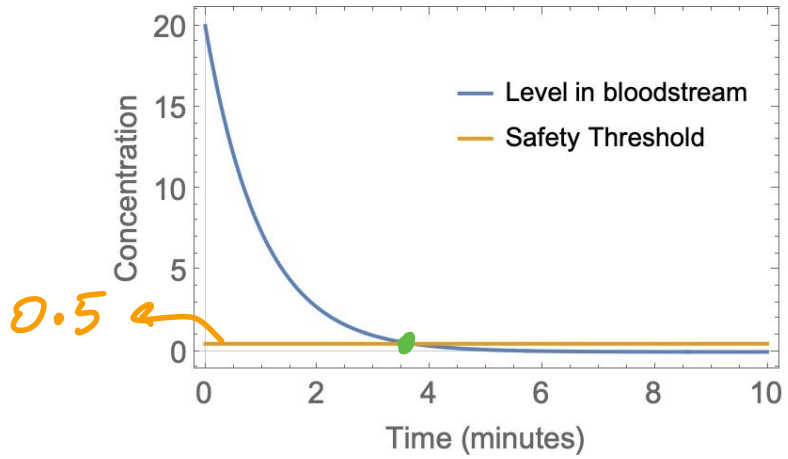


Analytical vs. Numerical Solutions: example

$f(x) = 20e^{-x}$

$f(x) = 0.5$

find 'root' of $g(x) \equiv f(x) - 0.5$



Analytical:

$$g(x) = 20e^{-x} - 0.5 = 0$$

$$20e^{-x} = 0.5$$

$$e^{-x} = \frac{1}{40}$$

$$-x = \log(1/40)$$

$$\boxed{x = \log 40 \approx \ldots}$$

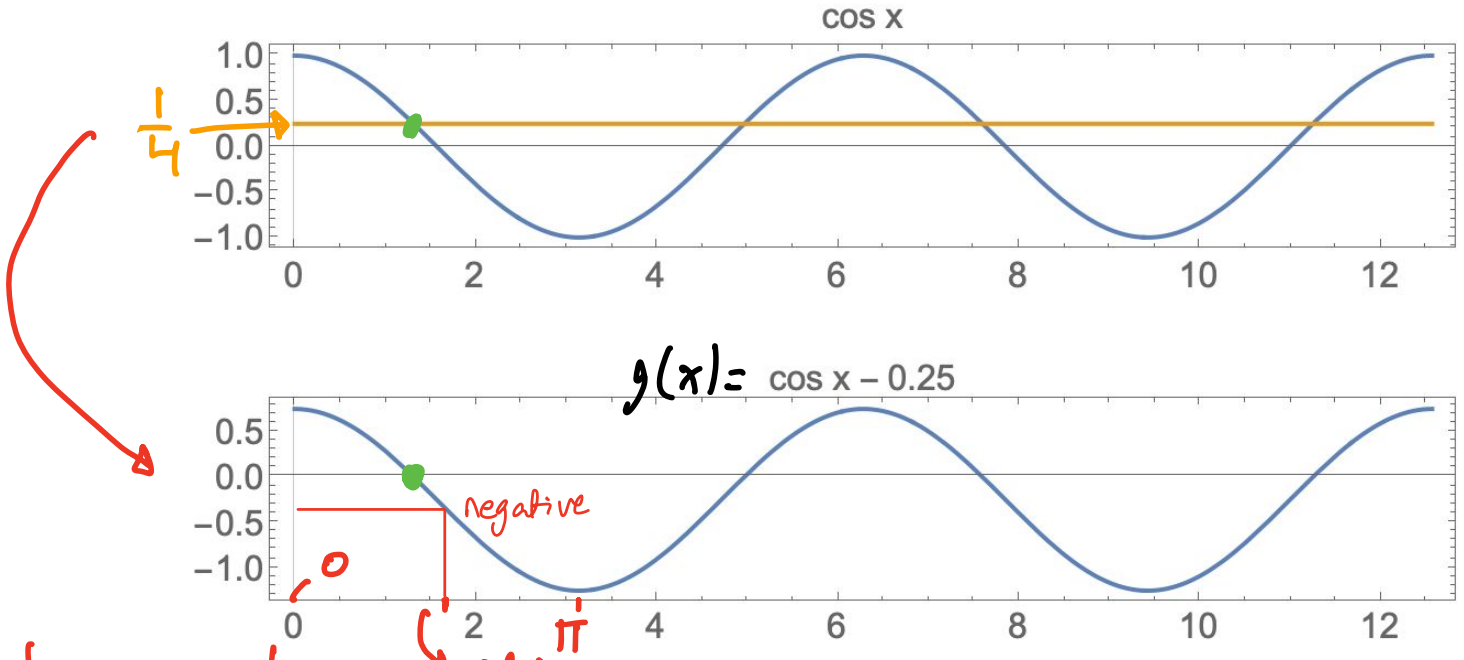
Computer:

Good at evaluating $g(x)$.

Note: sometimes, $g(x)$ is computationally expensive to evaluate.

Root-Finding Technique 1: the Bisection Algorithm

$$\cos(x) = \frac{1}{4}$$



Start with two numbers between which you know a solution exists.
sign of g must be diff. for the two ends.

→ find midpoint (here: π/2)

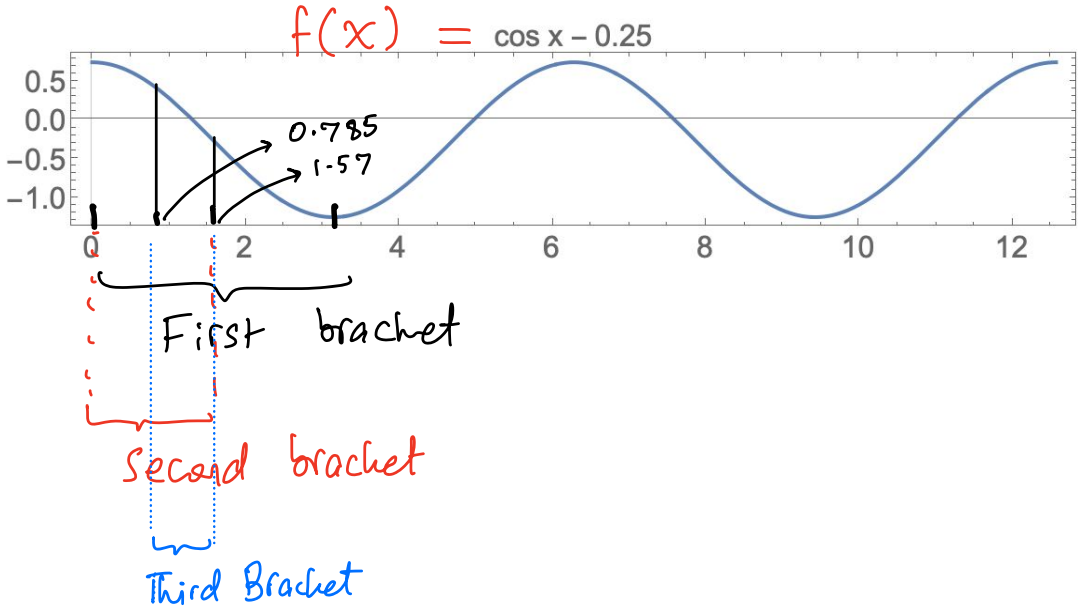
→ check sign of g(x) evaluated at the midpoint.

Procedure for Bisection Algorithm

1. Start with a bracket, i.e. two points between which you know a root exists.
   - Make sure that $\text{sgn}(f(x_1)) \neq \text{sgn}(f(x_2))$
2. Pick the midpoint
3. Decide which of the two regions must contain the root now.
4. Repeat until the size of bracket is small enough
   - Number of steps, or
   - Size of last bracket

Implementing the Bisection Algorithm by hand

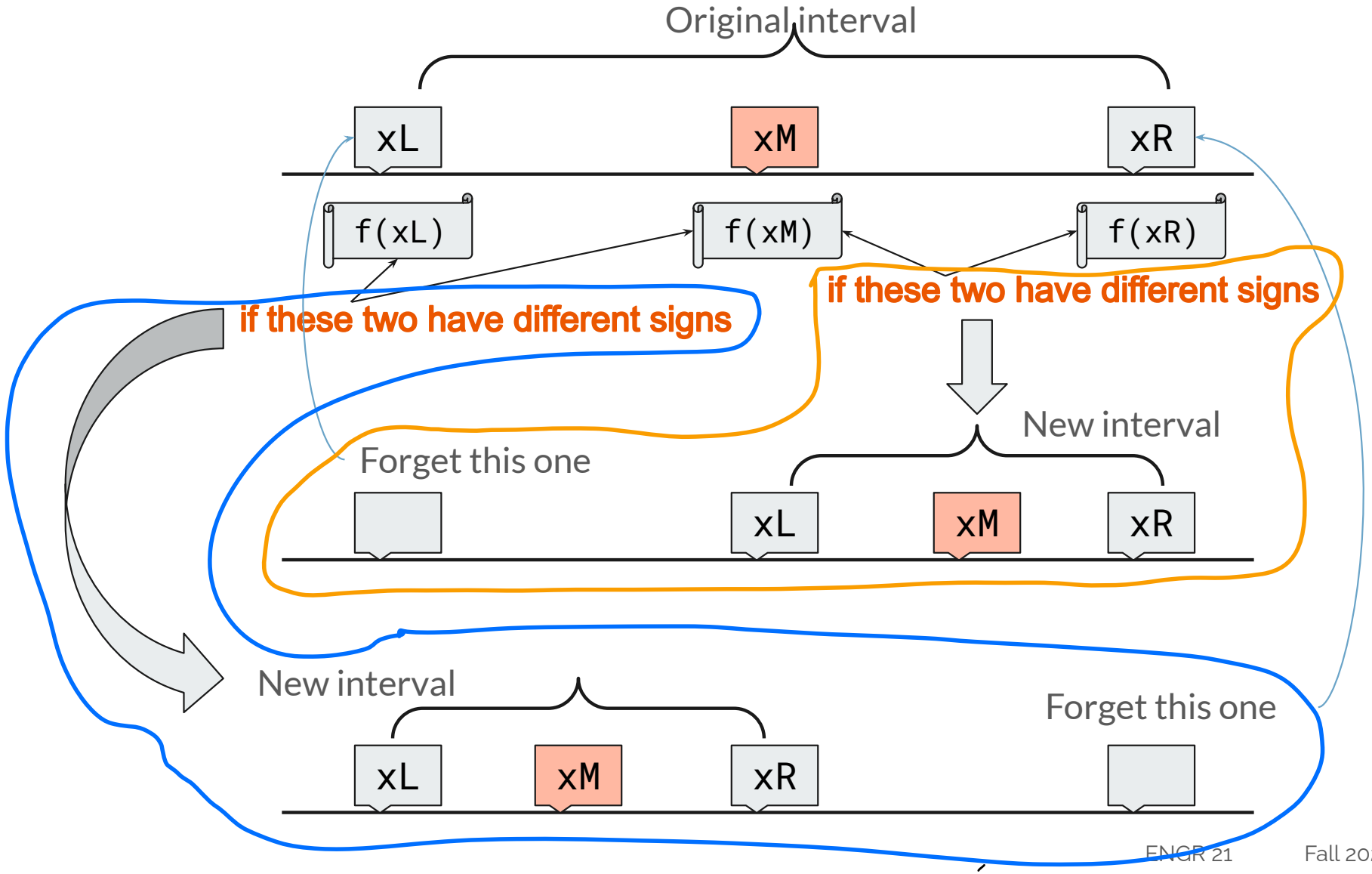
x_1	$f(x_1)$	x_2	$f(x_2)$
0	0.75	3.1415	−1.25
0	0.75	1.571	−0.25
0.785	0.457	1.571	−0.25



One step of the Bisection method

Try: $\cos(x) = 1/4$

$\sin^2 x = \frac{x}{2}$, find 3rd root.



Rate of Convergence of bisection method

$$\Delta x \rightarrow \frac{\Delta x}{2} \rightarrow \frac{\Delta x}{2^2} \rightarrow \frac{\Delta x}{2^3} \rightarrow \frac{\Delta x}{2^4} \quad \cdots \rightarrow \frac{\Delta x}{2^n}$$

How many steps will it take to reach an interval size of ε?

ε → how precisely you want to know. e.g. 10^{-6}

$$\varepsilon = \frac{\Delta x}{2^n} \quad \therefore \quad \log \varepsilon = \log \frac{\Delta x}{2^n}$$

$$n \simeq \log_2\left(\frac{\Delta x}{\varepsilon}\right)$$

1. Specify number of steps
   for loop
2. Specify a tolerance
   while loop

Python implementation

ENGR 21 Fall 2025

Resources

Bisection Algorithm

The following code *almost* implements the bisection algorithm. You must supply the code after the if-statements inside the while loop.

```
import math
from numpy import sign, cos, pi, exp, sin

def bisection(f,x1,x2,tol):
    # Carries out the bisection algorithm for the function f, using the
    # bracket (x1,x2) with a 'tolerance' of tol.
    f1 = f(x1)
    f2 = f(x2)
    if sign(f1) == sign(f2):
        # if the sign of f(x1) and f(x2) is the same,
        # there is probably no root between x1 and x2.
        print('Root is not bracketed between x1 and x2')
        return None

    er = 1 # some large value
    iters = 0
    while er > tol:
        iters+=1
        # uncomment the following line for debugging purposes
        # print(f'Root is between x1 = {x1:.3f} and x2 = {x2:.3f}')
        x3 = 0.5*(x1 + x2)
        f3 = f(x3)

        if f3 == 0.0:
            return x3
        if sign(f3) == sign(f2):
            # do something
            # new interval is ...
        else:
            # do something else
            # new interval is ...

        er = abs(x1-x2)
    return 0.5*(x1 + x2), iters

# Try it on the function 'cos'
bisection(cos,0.1,3.0,0.001)
```

a python function's name

left edge

right edge

Write code here.

- Complete function definition for bisection
- try it out on some other functions.

e.g.

```
def f1(x):
    return cos(x) - 0.25
```

e.g. find $x/2$ $\sin^2 x$