

HW 7

ENGR 21, Fall 2026.

October 29, 2026

Due Date	Thu, Oct 29
Turn in link	Gradescope PDF and Code
URL	emadmasroor.github.io/E21-F26/Homework/HW7

What to turn in

Combine all written HW into one PDF file.

Problem	Format
1	Code: One <code>.py</code> file containing all TFWR code.
2	Code: One or more <code>.py</code> files as well as image files
3	Code: One <code>.py</code> file
4.1,4.2	Code: one or more <code>.py</code> files
4.3	PDF
5	PDF

1 The Farmer Was Replaced

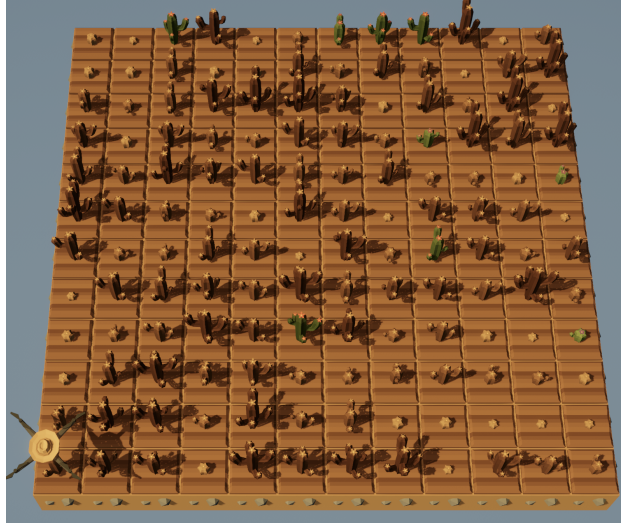
Download [the save file here](#) to ensure that you've unlocked all that's required for this assignment.

1.1 Sorting Cacti

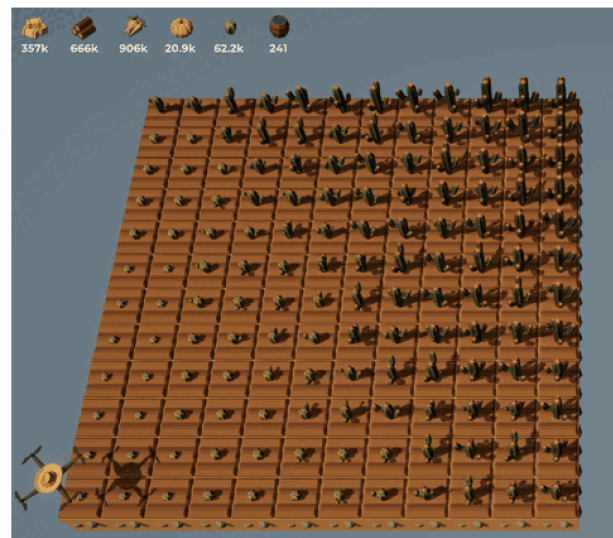
Cacti grow best when they are **sorted**. In TFWR, sorted cacti are those which are arranged in order of size relative to their neighboring cacti: the smallest cacti should be on the south and west, and largest cacti should be to the north and east.

You can check the size of the cactus underneath your drone using the function `measure()`, which returns an integer from 1 to 9. If you call `measure` with an argument, you can also measure the neighboring cacti (without moving your drone). For example, `measure(East)` measures the cactus to the East of the current location of the drone.

The `swap(<direction>)` function swaps the cactus underneath the drone for the cactus located `<direction>` away from the drone. For example, `swap(North)` swaps the cactus underneath the drone with a cactus to the north of the drone.



(a) Initially, cacti are randomly sized



(a) After sorting, correctly sized

Write a program that plants cacti on the entire farm, then sorts them into the correct order, then harvests them. It need only do this once.

i Hint

Your code should work ‘out of the box’, regardless of whether the farm is tilled or not. You can either

- use `clear()` and start from an untilled farm, and till each spot
- check the soil type and only `till()` if needed.

The precise sorting algorithm you use is not necessarily important. This is not a Computer Science class, so your algorithm does not need to be the most efficient one in the world, as long as it works correctly and accomplishes its objective.

i Success

You have succeeded when the entire farm of cacti can be harvested in a single ‘poof’, as seen in [?@fig-poof](#). Also notice that cacti are green when sorted and brown when unsorted.

i An example algorithm

On the class website you can see an example sorting algorithm in action in [?@fig-sort](#). Here, the drone works hard to sort the farm. It accomplishes its objective by **first** sorting each row, then sorting the columns.

You can also find this gif in video format hosted at [Panopto](#) at the following [link](#). This will allow you to have better control over the playback of the animation in [?@fig-sort](#).

2 Convergence of root-finding methods

The code given below compares the performance of three root-finding algorithms for finding a root (which we call x^*) of the function $f(x) = x^3 - 10x^2 + 5$ near $x = 8$. Each method is run for ten iterations. The numerical methods have not been correctly implemented, however, and the resulting figure is different from the desired one.

i True value

The true value of the root is approximately 9.949491057914386

Modify the code to accomplish the following objectives.

1. Correctly implement all three root-finding algorithms
2. Create a graph of $|x_n - x^*|$ against n , plotted on the usual linear axes. n should go up to 10.
3. Create a graph of $|x_n - x^*|$ against n , plotted on a logarithmic scale on the vertical axis. For this graph, n should go up to 50. See the documentation for [semilogy](#).

```
import numpy as np
import matplotlib.pyplot as plt

f2 = lambda x: x**3 - 10 * x**2 + 5
fp2 = lambda x: -20 * x + 3 * x**2

N = 10

x_newton = np.zeros(N)
x_secant = np.zeros(N)
x_bisect = np.zeros(N)

xold = 8

# Newton iterations
for i in range(N):
    xnew = xold - f2(xold)/fp2(xold)
    xnew = xold
    x_newton[i] = xnew

# Secant iterations
x1 = 8
x2 = 9
for i in range(N):
    m = (f2(x2)-f2(x1))/(x2-x1)
```

```

    c = f2(x1) - m*x1
    xnew = c/m
    x1 = xnew
    x_secant[i] = xnew

# Bisection iterations
xL = 8
xR = 11

for i in range(N):
    xM = (xL+xR)/2
    fxL = f2(xL)
    fxR = f2(xR)
    fxM = f2(xM)

    x_bisect[i] = xM
    if np.sign(xL) == np.sign(xM):
        # search in right side
        xL = xM
    elif np.sign(xR) == np.sign(xM):
        # search in left side
        xR = xM

# Plot x value against step
plt.plot(x_newton,marker="o")
plt.plot(x_secant,marker="s")
plt.plot(x_bisect,marker="*")
plt.legend(["Newton","Secant","Bisection"])
plt.title("$x_n$ vs $n$ for root of $x^3-10x^2+5$")
plt.xlabel("Iteration number $n$")
plt.grid(visible=True)
plt.savefig("test.png")
plt.show()

```

3 A Python program for the Secant method

Note

Combine all code for this problem into a single file.

3.1 Write a function

Write a Python function that takes as input the following 4 arguments, in order:

1. A function f , whose root is to be found.
2. x_1 , one estimate of the root
3. x_2 , another estimate of the root
4. n , the number of steps of the secant method that are to be carried out.

It should output a guess for the root of $f(x)$.

Functions as arguments

A Python function can be an argument to another function. For example, the following function takes as input the independent variable x and the function g , and returns $g^2(x) = g(x) \times g(x)$

```
from numpy import sin

def square(func,xval):
    return func(xval)*func(xval)

print(f"{square(sin,2):.3f}")
```

3.2 Use your function

Use your function to determine a root of the function $\sin e^{x/2}$ starting from $x = 1$ and $x = 2$ as your initial guesses. For this part, your code should call the function you wrote in Section 3.1 and should print a value for the root.

4 A Python program for Newton's method

4.1 Write a function

Write a Python function for Newton's method that accepts as argument the following four entities:

1. The function $f(x)$ whose root is to be found,
2. The function $f'(x)$, its derivative,
3. A guess x_0 from which to start calculations, and
4. A tolerance parameter ε .

Given a function f and its derivative f' , your Python function should apply Newton's method to find a root of $f(x)$ starting from $x = x_0$. It should continue searching for a root until the following termination criterion has been met. After termination, it should return the latest guess of the root's value.

Termination criterion

Terminate the search for a root when

$$\frac{|x_{n+1} - x_n|}{|x_n|} < \varepsilon,$$

where ε is a tolerance parameter that is generally a small number.
A smaller ε will lead to a more accurate solution.

4.2 Use your function

Use your function to find a root of $f(x) = \sin e^{x/2}$, starting from (1) $x_0 = 2$ and (2) $x_0 = 1$. Write print statements to report to a user the results of your calculations.

4.3 Explain its behavior

By using a graph of the function $\sin e^{x/2}$, explain the results from Section 4.2. Why does the program find the results it does? Sketch a few steps of Newton's method by hand to aid in your explanation.

5 Graphical Illustration of Root-finding

5.1 Bisection Method for $f(x) = (x + \pi)(x - \pi/2)(x - 5)$

5.1.1 Bisection Method with numbers

A root of the function $f(x) = (x + \pi)(x - \pi/2)(x - 5)$ is to be found using the bisection method starting from the bracket $[-4, -3]$. Use the following table to record the results of the bisection method for six steps. Some steps have been done for you.

$x_L,$	$x_M,$	$x_R]$	$[f(x_L), f(x_M), f(x_R)]$	Next bracket
$[-4.0000000, -3.5000000, -3.0000000]$			$[-43.0, -15.4, +5.18]$	right bracket
			$[-15.4, -4.31, +5.18]$	left bracket

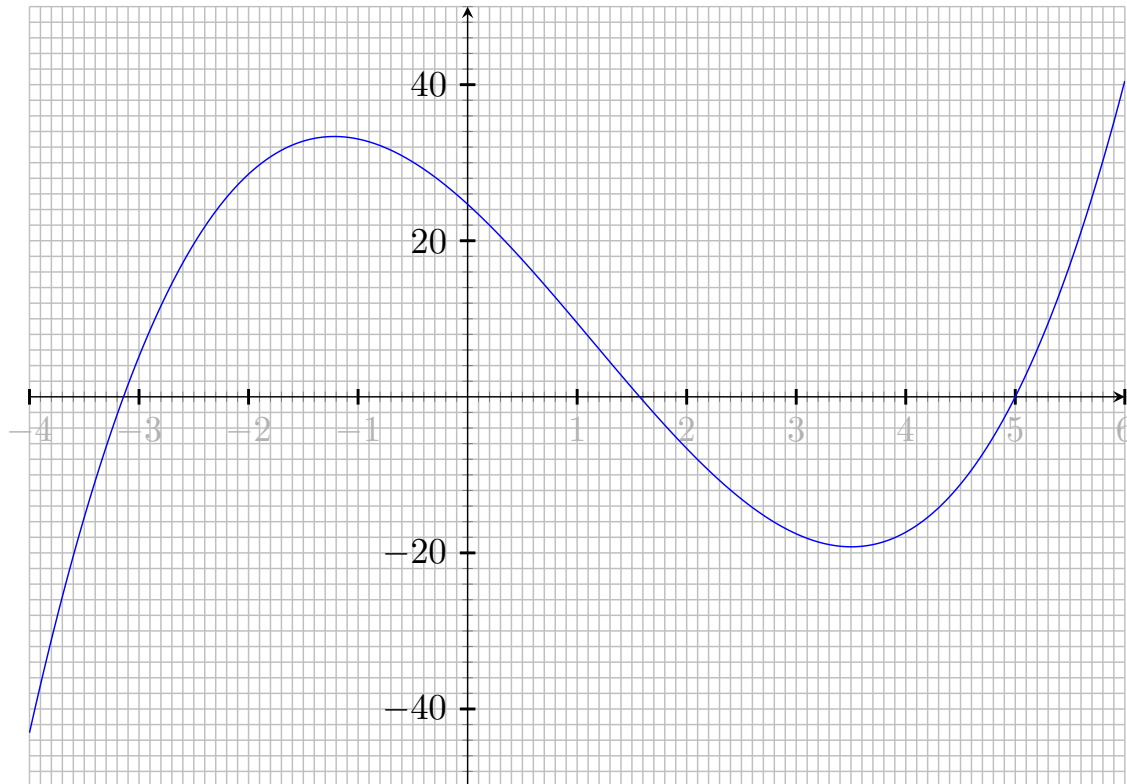
5.1.2 Bisection method with pictures

Using hand-calculations and a simple calculator, explain how the bisection algorithm will behave when initialized with each of the following brackets. For each case, indicate whether or not it will converge on a solution, and if so, which root it will converge on.

- $[-4, -2]$
- $[-4, +2]$

- $[-4, +6]$
- $[0, +4.9]$
- $[+2, +4]$

You may use the following diagram to solve this problem.



5.2 Newton's Method

Use Newton's method to find the root of the function $x^2 - 2x - 4$, starting from the estimate that the root is at $x_0 = 0$. Find the next two estimates according to Newton's method.

