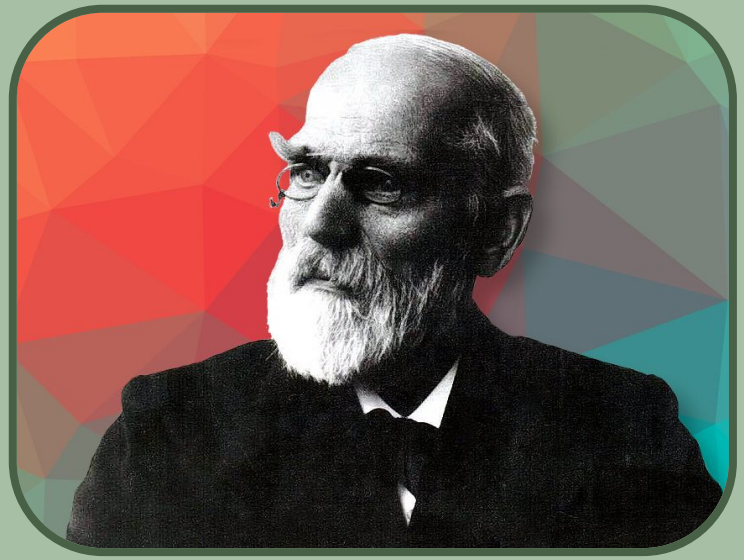
# Root Finding: Newton's Method vs. Secant Method

## Using van der Waals Equation for Comparison

Tianque Cao, Reese Karo, Vidushi Mittal,
Xiao Yang, Shangcao Yuan

# 100th Anniversary: Death of Johannes Diderik van der Waals

Born on November 23, 1837, in Leiden in The Netherlands



From 1862 to 1908, he taught Physics and Mathematics in several universities found in Amsterdam

The Nobel Prize Winner in Physics in 1910 for his work on the equation of state for gases and liquids

Passed away in Amsterdam on March 8, 1923

# Newton's Method and Secant Method: *Intuition*

## Newton's Method

Newton's Method begins with an initial guess and uses the zero at the tangent line of the initial guess to start the next iteration

## Secant Method

This method begins with two initial guesses and instead of a derivative, it uses the root of their secant line to start the next iteration

# Newton's Method and Secant Method: *Algorithm*

## Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

## Secant Method

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

# What is the van der Waals Equation?

## The Ideal gas Law:

$$pV = nRT$$

It describes the behavior of an idealized gas under different conditions. However, this equation neglects intermolecular interactions between the molecules in a gas, as well as the finite sizes of the molecules.

## Van der Waals Equation:

$$n \cdot R \cdot T = \left(P + \frac{a \cdot n^2}{V^2}\right)(V - b \cdot n)$$

The Van der Waals equation includes the effects of the intermolecular interactions and non-zero molecular sizes. It describes experimentally observed processes that the ideal gas law cannot account for.

# What is the van der Waals Equation?

## Variables:

**V** = Volume in Liters
**P** = Pressure **(120 ATM)**
**T** = Temperature at pressure **P (696 K)**
**a, b** = van der Waals constants
- ➔ **a**: measure for molecule average attraction **(3.61 $L^2$/ATM/$mol^2$)**
- ➔ **b**: adjusts for volume occupied by particle**(0.0428 L/mol)**

**R** = Universal gas constant ≈ **.0821 L*ATM/(Mol * K)**

## Equation:

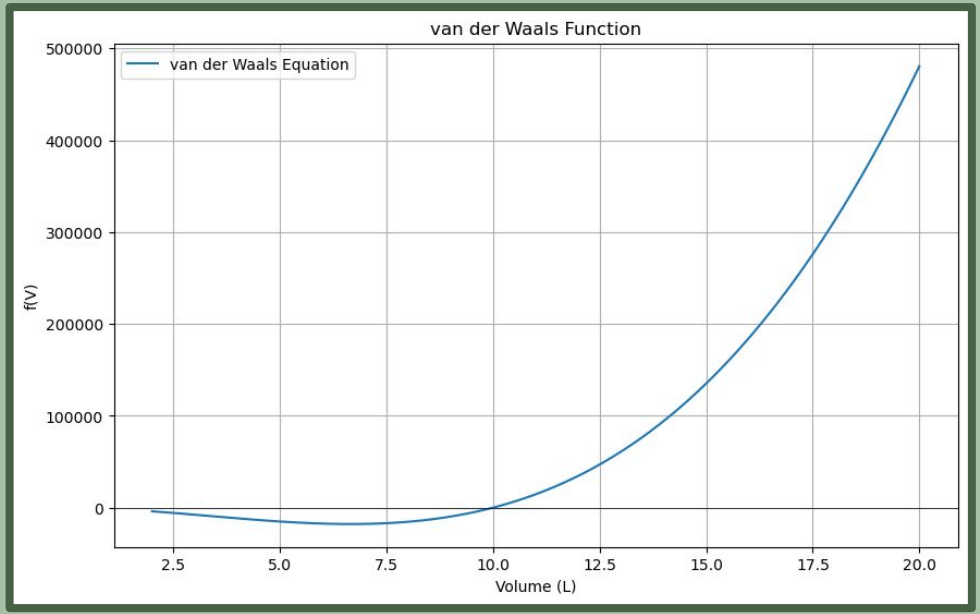
$$n \cdot R \cdot T = \left(P + \frac{a \cdot n^2}{V^2}\right)(V - b \cdot n)$$

## Our Project:

By subtracting n*R*T on both sides, we can solve for f(V) using our root solving methods

# van der Waals function: *Graph*

- We are trying to find the volume, **V** in Liters, without the knowledge of the graph we have to our right.
- In other words, **what quantity of volume will make f(V) = 0**. Our methods will help us find this exact value



```python
import numpy as np

def newtons_method(f, f_prime, x0, tol=10e-10, max_iterations=100, verbose=False):
    """
    Applies Newton's method to find the root of the equation f(x) = 0.
    """
    x = x0
    converged = False

    for i in range(max_iterations):
        fx = f(x)
        fpx = f_prime(x)

        # Check if derivative is zero
        if fpx == 0:
            raise ValueError(f"Derivative is zero at V = {x}.")

        # Update x
        x_new = x - fx / fpx

        # Check convergence criteria
        if np.abs(f(x_new)) < tol and np.abs(x_new - x) / np.abs(x) < tol:
            converged = True
            x = x_new
            break

        x = x_new

        if verbose:
            true_x = 10 #Liters
            error = np.abs(true_x - x)
            print(f"Iteration {i}: V = {x}, f(V) = {fx}, error (true x - x), {error}")

    # Final convergence check
    if not converged:
        print("Warning: Newton's method did not converge after the maximum number of iterations.")

    return x, i + 1, converged
```

# Newton's Method: Code

- **Input parameters:** Requirements for our code is a function f and its derivative,, an initial guess x0, a tolerance, and maximum iteration
- **Continuous iteration**: By finding the value where the tangent line reaches the x-axis, we continuously iterate until a stopping point is reached
- **Error warnings:** An error is raised if our derivative reaches a value equal to 0

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

```python
import numpy as np

def secant_method(f, x0, x1, tol, max_iterations, verbose=False):
    '''
    Uses the Secant method to find a root of a function.

    Parameters:
    - f: The function for which to find the root.
    - x0, x1: The initial two guesses for the root.
    - tol: The tolerance for the convergence of the method.
    - max_iterations: The maximum number of iterations to perform.
    - verbose: Whether to print the process (default: False).

    Returns:
    - A dictionary with the root, the number of iterations, and a flag indicating convergence.
    '''

    if tol <= 0:
        raise ValueError("Tolerance must be positive.")

    for i in range(max_iterations):
        f_x0 = f(x0)
        f_x1 = f(x1)
        denominator = (f_x1 - f_x0)

        # Avoid division by zero
        if denominator == 0:
            print("Warning: Division by zero encountered in the Secant method.")
            return {"root": None, "iterations": i, "converged": False}

        x_new = x1 - (f_x1 * (x1 - x0) / denominator)

        if verbose:
            true_x = 10
            error = true_x - x_new
            print(f"Iteration {i+1}: x = {x_new}, f(x) = {f(x_new)}, error: {error}")

        if np.abs(x_new - x1) < tol:
            return {"root": x_new, "iterations": i + 1, "converged": True}

        x0, x1 = x1, x_new

    return {"root": x1, "iterations": max_iterations, "converged": False}
```

# Secant Method: Code

- **2 initial guesses:** In the secant method, we start with two initial guesses to then compute the secant line that crosses the x axis, where find the next value $x_{n+1}$

- **Iterative computation:** These steps repeat until converging around the value which makes f(V) = 0

- **Dictionary use:** In this particular case, we have saved the results into a dictionary to show different methods are possible for retrieving values

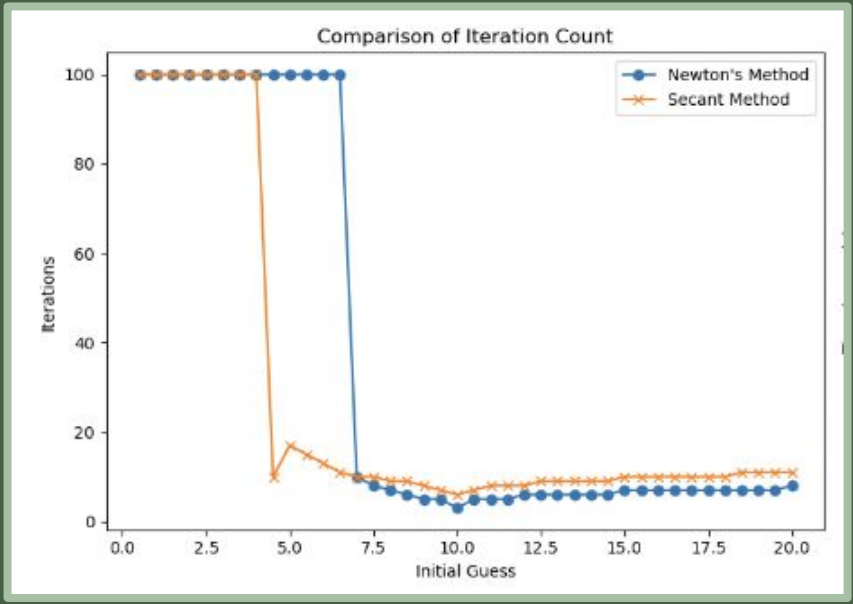
$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

# Method Comparison

| | Function evaluations per iteration | Runtime (seconds) | Error: True root - calculated root |
|---|---|---|---|
| Newton's: (x0 = 15) | **2** | 0.0001161098480224609 4 | 0.013241528133654867 |
| Secant: (x0, x1 = 10, 15) | **2** | 0.0003099441528320312 5 | 0.013241528133653091 |

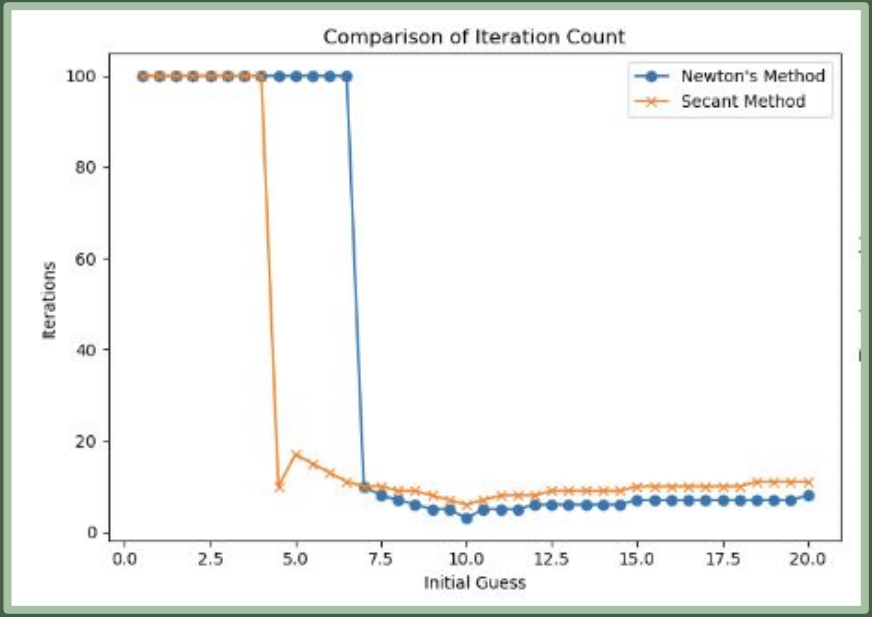
The results of running both methods. We will delve into the comparisons in the next slides

# Comparing the methods results!



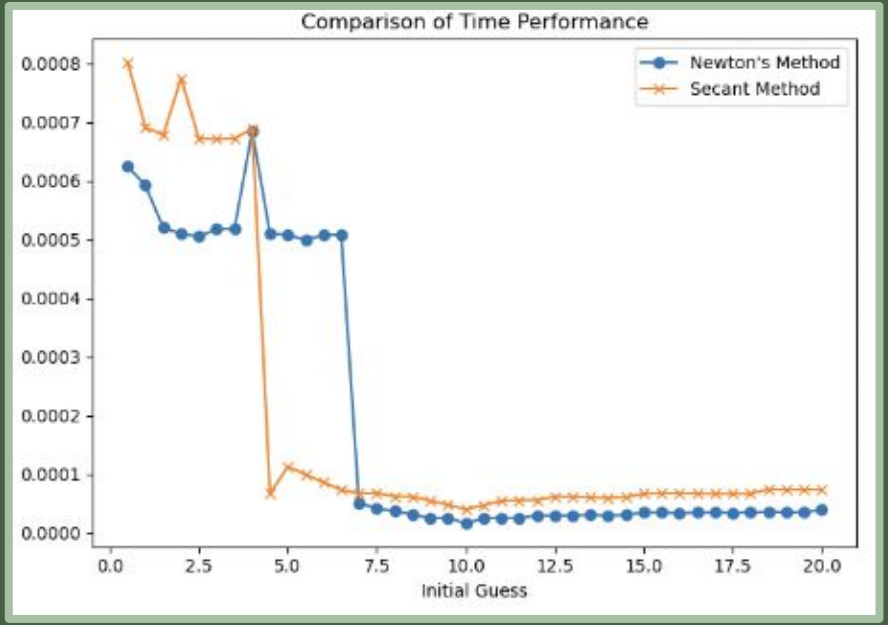

# Iteration Count: Newton's vs Secant Methods

- Here we can see the iteration count needed for each method based on the initial guesses.
- For secant method: We see that when we have initial guesses $x_0$ in the range of (.5, 5) and $x_1$ in range (10.5, 15) we see the methods do not converge
- However we notice a change in performance when the initial guesses range from $x_0$ in (5,20) and $x_1$ in (15, 30)




# Iteration Count: Newton's vs Secant Methods

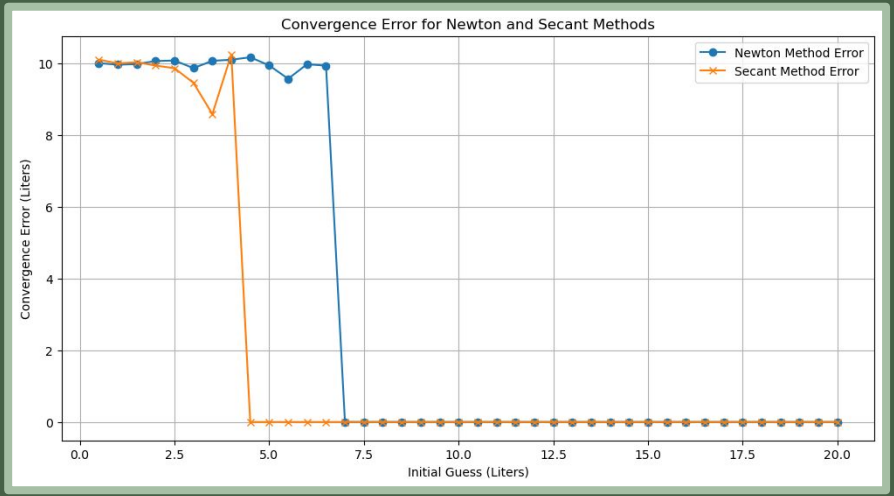
- For Newton's method, we can see for initial guesses less than roughly 7.5, the method does not converge (because we set a max iteration tolerance at 100 steps)
- For initial guesses greater than 7.5, we have a convergence rate that is roughly the same until we reach $x_0$ = 20.
- We notice that the blue line has a global minimum around 10, which tells us that a guess of 10 converges the quickest! (not necessarily the most accurate, see next few slides)

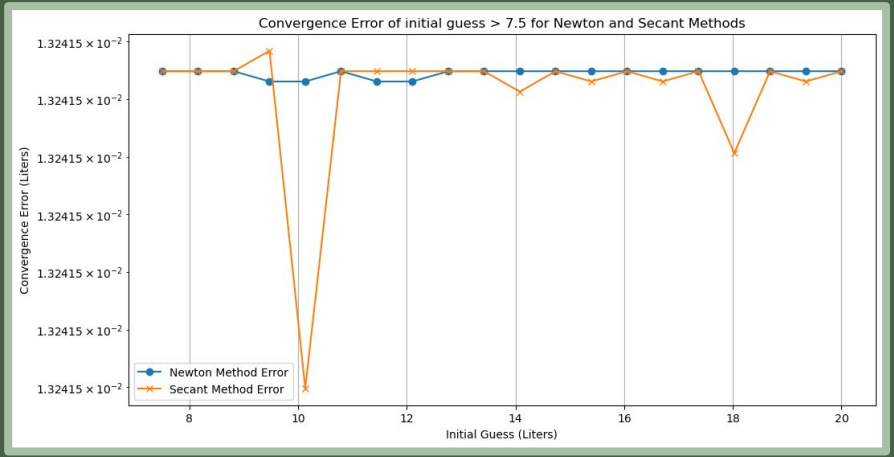



# Runtime Count: Newton's vs Secant Methods

- Note that this graph will vary with each run, but we can draw conclusions and note the general trends.

- We measure the runtime as the time it takes for the iteration error to be under $10^{-8}$. Notice here that regardless of the initial guess, run time of secant method is longer than Newton's, unless the initial guess is between approx. 5 and 7.




# Error Analysis: Newton vs. Secant method

- In the graph on top, we are comparing the convergence errors of the final value of V that makes f(V) versus the true value of the Volume.
- In the graph below we can look closer at the values seen in the top graph for initial guesses greater than 7.5
- Analysis: The secant method with an initial guess of $x_0 = 10$ and $x_1 = 20$, gives us the best convergence error between that and the Newton's method.

# Questions ?