Numerical Methods
Lab 6. Root Finding

Maryna Borovyk

November 6, 2024

1 Introduction

The velocity of water v discharged from a cylindrical tank through a long pipe can be computed using the equation:

$$v(t) = \sqrt{2gH}\tanh\left(\frac{\sqrt{2gH}}{2L}t\right) \tag{1}$$

where $g = 9.81\,\mathrm{m/s}^2$, H is the initial height (m), L is the pipe length (m), and t is the elapsed time (s). The objective is to determine the initial height H needed to achieve $v = 4\,\mathrm{m/s}$ in 3 seconds for a 5-meter long pipe using graphical, bisection, and another numerical method.

[118]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[119]:
```
# Constants
g = 9.81   # acceleration due to gravity (m/s^2)
L = 5      # pipe length (m)
v_target = 4   # target velocity (m/s)
t = 3      # elapsed time (s)
```

[120]:
```
def velocity(H):
    return np.sqrt(2 * g * H) * np.tanh(np.sqrt(2 * g * H) * t / (2 * L))
```

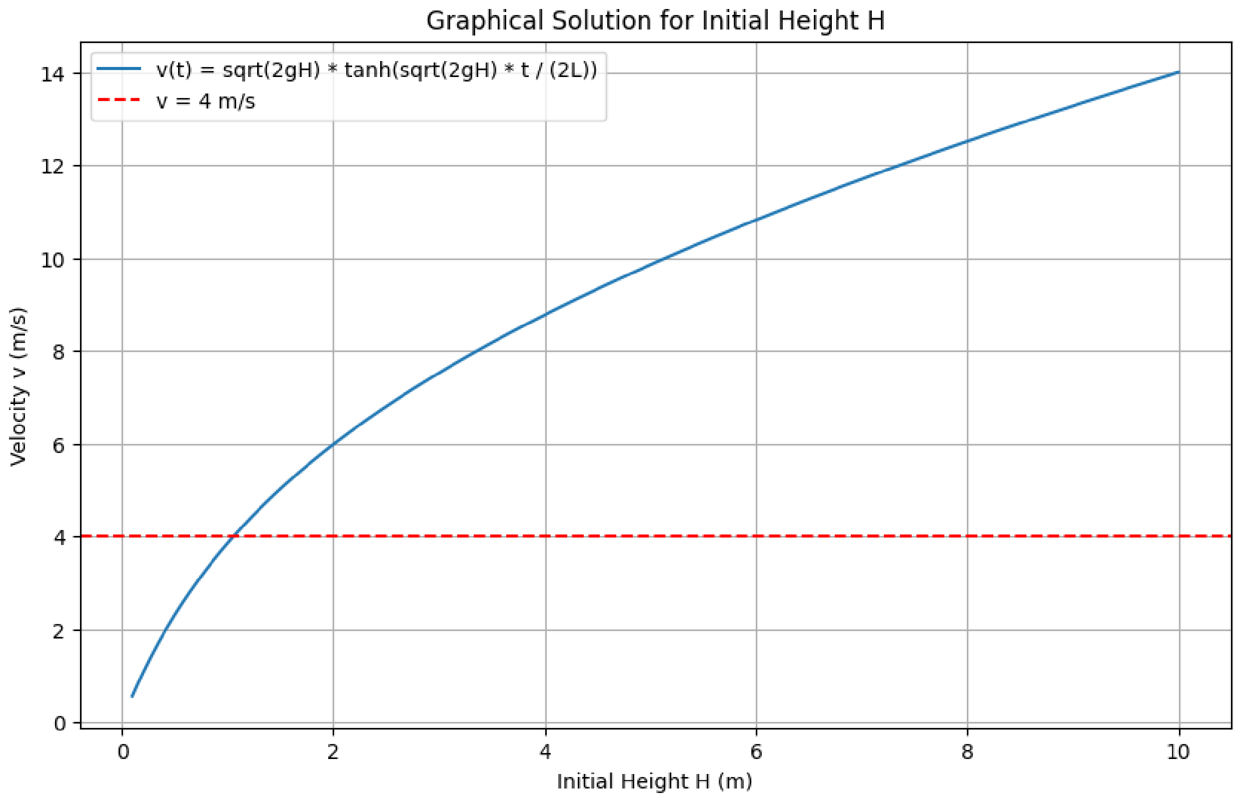
2 Task (a):Graphical Method

The graphical method involves plotting the velocity function and visually identifying the initial height H that corresponds to the target velocity. The following Python code was used to generate the plot:

[121]:
```
H_values = np.linspace(0.1, 10, 400)
v_values = [velocity(H) for H in H_values]

plt.figure(figsize=(10, 6))
plt.plot(H_values, v_values, label='v(t) = sqrt(2gH) * tanh(sqrt(2gH) * t /␣
 ↪(2L))')
plt.axhline(y=v_target, color='r', linestyle='--', label='v = 4 m/s')
plt.xlabel('Initial Height H (m)')
```

```python
plt.ylabel('Velocity v (m/s)')
plt.title('Graphical Solution for Initial Height H')
plt.legend()
plt.grid(True)
plt.show()
```



3 Task (b): Bisection Method

The bisection method is an incremental search method that repeatedly divides the interval in half to find the root. The following Python code implements the bisection method:

[122]:
```python
def bisection(xl, xu, f):
    while xu - xl > 0.00001:
        xr = (xl + xu) / 2

        if f(xr) == 0:
            break

        if f(xl) * f(xr) < 0:
            xu = xr
        else:
            xl = xr
    return xr
```

[123]:
```python
def f(H):
    return velocity(H) - v_target
```

[124]:
```python
H_sol = bisection(0.1, 10, f)
print(f'The initial height H is {H_sol:.4f} m')
```

```
The initial height H is 1.0580 m
```

4 Task (c): Newton-Raphson Method, scipy bisect method

The Newton-Raphson method is an iterative root-finding method that uses the derivative of the function.

[125]:
```python
from scipy.optimize import bisect
```

[126]:
```python
H_sol2 = bisect(f, 0.1, 10)
print(f'The initial height H is {H_sol2:.4f} m')
```

```
The initial height H is 1.0580 m
```

[127]:
```python
def velocity_derivative(H):
    term1 = np.sqrt(2 * g / H) * np.tanh(np.sqrt(2 * g * H) * t / (2 * L))
    term2 = (2 * g * H) / (2 * L) * (1 / np.cosh(np.sqrt(2 * g * H) * t / (2 *␣
↪L)))**2
    return term1 + term2
```

[128]:
```python
def newton_raphson(H0, f, f_prime, tol=1e-5, max_iter=100):
    H = H0
    while max_iter > 0:
        H_new = H - f(H) / f_prime(H)

        if abs(H_new - H) < tol:
            return H_new

        H = H_new
        max_iter -= 1
    raise ValueError("Newton-Raphson method did not converge")
```

[129]:
```python
H_initial_guess = 1.0
H_sol3 = newton_raphson(H_initial_guess, f, velocity_derivative)
print(f'The initial height H using Newton-Raphson is {H_sol3:.4f} m')
```

```
The initial height H using Newton-Raphson is 1.0580 m
```

[129]:
```python
```

5 Results and Observations

The graphical method provided a visual estimate of the initial height H. The bisection method and Newton-Raphson method both converged to an initial height H of approximately 1.0580 meters. The results are consistent across different methods, demonstrating the reliability of the numerical approaches.

6 Conclusion

The initial height H required to achieve a velocity of 4 m/s in 3 seconds for a 5-meter long pipe was determined using graphical, bisection, and Newton-Raphson methods. All methods provided consistent results, with $H \approx 1.0580$ meters. The bisection and Newton-Raphson methods are effective numerical techniques for solving root-finding problems.