Applied Mechanics Tutorial

A Concise Introduction to Computational Tools

Contents

List of Figures

List of Tables

Preface

A Concise Introduction to Computational Tools provides a series of tutorials derived from lecture notes, offering clear and focused guidance on essential topics. No prior programming experience is required.

Students are expected to be comfortable with college-level mathematics and science and are strongly encouraged to consult reliable reference texts. The tutorials also assume working familiarity with either macOS or Microsoft Windows.

For best results, these materials should be used on a laptop or desktop computer rather than a tablet or smartphone.

Recommended Books

Hannah & Hillier (1995)

Russell et al. (2021)

Bolton (2021)

Shaw (2017)

Sweigart (2021)

Sweigart (2020)

Dedication

To my students, whose inspiration and encouragement made this work possible.

Attributions

This book incorporates figures, data, and other resources from the following sources:

- Figures in the formulae, the centroids and second moments of area, are adapted from the Wikipedia pages on Centroids and Second moments of area.

Acknowledgements

I would like to thank the colleagues and friends who helped shape this project.
Special thanks to their valuable feedback and support.
Any errors or omissions are my own.

Study Guide

- First and foremost, solve the problem sets using only pen, paper, and a calculator. After completing a problem by hand, slightly modify the conditions or variables and solve it again. Then, input the new values into the Python script and compare the results with your hand-calculated solution.
- Buddy system: Study with a classmate. Teaching and helping one another significantly deepens understanding of the material. Students are strongly encouraged to work through problem sets collaboratively. For example, one student solves the odd-numbered problems while the other solves the even-numbered ones; then each explains their solutions to the other.
- Practice, practice, practice: As the saying goes, "practice makes perfect." More accurately in this context, "good practice makes perfect." That means pen-and-paper-first practice. Use Python as a verification tool, not a crutch. Always derive the solution analytically on paper first. Only after obtaining a complete hand-calculated result should you run the corresponding Python script. This approach echoes a proverb commonly attributed to Confucius:

  I hear and I forget

  I see and I remember

  I do and I understand
- Maintain a solution notebook: For each major problem, keep a single document (physical or digital) containing:
  - The full hand derivation
  - Key numerical results with units
  - A brief paragraph summarizing insights or reconciling any intermediate discrepancies

  This record becomes an invaluable review resource before exams.

Python Tutorial

This tutorial introduces Python programming, covering basic concepts with examples to illustrate key points. We will start by using Python as a calculator, then explore variables, functions, and control flow.

Requirements

To follow this tutorial, the easiest way to get started is by using a web-based Python environment. This lets you write and run Python code right in your browser; no downloads or setup needed. I recommend python-fiddle.com, an easy-to-use online editor that lets you experiment with Python instantly and solve your problem sets effortlessly.

If you prefer working on your own computer, make sure you have Python (version 3.10 or later) installed. Python works on Windows, macOS, and Linux. You'll also need a text editor or an Integrated Development Environment (IDE) to write your code. I recommend Positron, a beginner-friendly IDE with a built-in terminal, though other editors like VS Code or PyCharm are also good options.

Basic Syntax

Python uses indentation (typically four spaces) to define code blocks. A colon (`:`) introduces a block, and statements within the block must be indented consistently. Python is case-sensitive, so `Variable` and `variable` are distinct identifiers. *Statements typically end with a newline, but you can use a backslash (`\`) to continue a statement across multiple lines.*

```
total = 1 + 2 + 3 + 4 + 5
print(total)  # Output: 15
```

Basic syntax rules:

- Comments start with `#` and extend to the end of the line.

- Strings can be enclosed in single quotes (`'`), double quotes (`"`), or triple quotes (`'''` or `"""`) for multi-line strings.
- Python is case-sensitive, so `Variable` and `variable` are different identifiers.

The `print()` Function

The `print()` function displays output in Python.

```
name = "Rudolf Diesel"
year = 1858
print(f"{name} was born in {year}.")
```

Output: `Rudolf Diesel was born in 1858.`

Formatting in `print()`

The following table illustrates common f-string formatting options for the `print()` function:

Format	Code	Example	Output
Round to 2 decimals	`f"{x:.2f}"`	`print(f"{3.14159:.2f}")`	`3.14`
Round to whole number	`f"{x:.0f}"`	`print(f"{3.9:.0f}")`	`4`
Thousands separator	`f"{x:,.2f}"`	`print(f"{1234567.89:,.2f}")`	`1,234,567.89`
Percentage	`f"{x:.1%}"`	`print(f"{0.756:.1%}")`	`75.6%`
Currency (CDN)	`f"${x:,.2f}"`	`print(f"${1234.5:,.2f}")`	`$1,234.50`
Currency (EUR)	`f"€{x:,.2f}"`	`print(f"€{1234.5:,.2f}")`	`€1,234.50`
Currency (JPY)	`f"¥{x:,.0f}"`	`print(f"¥{1234567:,.0f}")`	`¥1,234,567`

Variables and Data Types

Variables store data and are assigned values using the `=` operator.

```
x = 10
y = 3.14
name = "Rudolph"
```

Python has several built-in data types, including:

- Integers (`int`): Whole numbers, e.g., `10`, `-5`
- Floating-point numbers (`float`): Decimal numbers, e.g., `3.14`, `-0.001`
- Strings (`str`): Text, e.g., `"Hello"`, `'World'`
- Booleans (`bool`): `True` or `False`

Arithmetic Operations

```
a = 10
b = 3
print(a + b)  # Addition: 13
print(a - b)  # Subtraction: 7
print(a * b)  # Multiplication: 30
print(a / b)  # Division: 3.3333...
print(a // b) # Integer Division: 3
print(a ** b) # Exponentiation: 1000
```

String Operations

```
first_name = "Rudolph"
last_name = "Diesel"
full_name = first_name + " " + last_name  # Concatenation using +
print(full_name)  # Output: Rudolph Diesel
print(f"{first_name} {last_name}")  # Concatenation using f-string
print(full_name * 2)  # Repetition: Rudolph DieselRudolph Diesel
print(full_name.upper())  # Uppercase: RUDOLPH DIESEL
```

Note: String repetition (`*`) concatenates the string multiple times without spaces. For example, `full_name * 2` produces `Rudolph DieselRudolph Diesel`.

Python as a Calculator in Interactive Mode

Python's interactive mode allows you to enter commands and see results immediately, ideal for quick calculations. To start, open a terminal (on macOS, Linux, or Windows) and type:

```
python3  # Use 'python' on Windows if 'python3' is not recognized
```

You should see the Python prompt:

```
>>>
```

Enter expressions and press **Enter** to see results:

```
2 + 3  # Output: 5
7 - 4  # Output: 3
6 * 9  # Output: 54
```

```
8 / 2  # Output: 4.0
8 // 2 # Output: 4
2 ** 3 # Output: 8
```

Parentheses for Grouping

```
(2 + 3) * 4  # Output: 20
2 + (3 * 4)  # Output: 14
```

Variables

```
x = 10
y = 3
x / y  # Output: 3.3333333333333335
```

Exiting Interactive Mode

To exit, type:

```
exit()
```

Alternatively, use: - **Ctrl+D** (macOS/Linux) - **Ctrl+Z** then Enter (Windows)

Control Flow

Control flow statements direct the execution of code based on conditions.

Conditional Statements

Conditional statements allow you to execute different code blocks based on specific conditions. Python provides three keywords for this purpose:

- `if`: Evaluates a condition and executes its code block if the condition is `True`.
- `elif`: Short for "else if," it checks an additional condition if the preceding `if` or `elif` conditions are `False`. You can use multiple `elif` statements to test multiple conditions sequentially, and Python will execute the first `True` condition's block, skipping the rest.
- `else`: Executes a code block if none of the preceding `if` or `elif` conditions are `True`. It serves as a fallback and does not require a condition.

The following example uses age to categorize a person as a Minor, Adult, or Senior, demonstrating how `if`, `elif`, and `else` work together.

```
# Categorize a person based on their age
age = 19
if age < 18:
    print("Minor")
elif age <= 64:
    print("Adult")
else:
    print("Senior")
```

Output: `Adult`

For Loop

A `for` loop iterates over a sequence (e.g., list or string).

```
components = ["piston", "liner", "connecting rod"]
for component in components:
    print(component)
```

Output:

```
piston
liner
connecting rod
```

While Loop

A `while` loop executes as long as a condition is true. Ensure the condition eventually becomes false to avoid infinite loops.

```
count = 0
while count <= 5:
    print(count)
    count += 1
```

Output:

```
0
1
2
3
4
5
```

Functions

The `def` Keyword

Functions are reusable code blocks defined using the `def` keyword. They can include default parameters for optional arguments.

```
def add(a, b=0):
    return a + b
print(add(5))      # Output: 5
print(add(5, 3))   # Output: 8

def multiply(*args):
    result = 1
    for num in args:
        result *= num
    return result
print(multiply(2, 3, 4))  # Output: 24
```

The `lambda` Keyword

The `lambda` keyword creates anonymous functions for short, one-off operations, often used in functional programming.

```
celsius_to_fahrenheit = lambda c: (c * 9 / 5) + 32
print(celsius_to_fahrenheit(25))  # Output: 77.0
```

The `math` Module

The `math` module provides mathematical functions and constants.

```
import math
print(math.sqrt(16))  # Output: 4.0
print(math.pi)        # Output: 3.141592653589793
```

```
import math
angle = math.pi / 4  # 45 degrees in radians
print(math.sin(angle))  # Output: 0.7071067811865475 (approximately √2/2)
print(math.cos(angle))  # Output: 0.7071067811865476 (approximately √2/2)
print(math.tan(angle))  # Output: 1.0
```

Note: Floating-point arithmetic may result in small precision differences, as seen in the `sin` and `cos` outputs.

```
import math
print(math.log(10))       # Natural logarithm of 10: 2.302585092994046
print(math.log(100, 10))  # Logarithm of 100 with base 10: 2.0
```

Converting Between Radians and Degrees

The `math` module provides `math.radians()` to convert degrees to radians and `math.degrees()` to convert radians to degrees, which is useful for trigonometric calculations.

> **Warning**
>
> In Python, the math module's trigonometric functions (sin, cos, tan, etc.) expect angles in radians, not degrees. Always convert degrees to radians before using sin, cos, or tan in Python's math module.

```
import math
degrees = 180
radians = math.radians(degrees)
print(f"{degrees} degrees is {radians:.3f} radians")  # Output: 180 degrees is 3.142 radians

radians = math.pi / 2
degrees = math.degrees(radians)
print(f"{radians:.3f} radians is {degrees:.1f} degrees")  # Output: 1.571 radians is 90.0 degrees
```

Writing Python Scripts

Write Python code in a `.py` file and run it as a script. Create a file named `script.py`:

```
# script.py
import math
print("Square root of 16 is:", math.sqrt(16))
print("Value of pi is:", math.pi)
print("Sine of 90 degrees is:", math.sin(math.pi / 2))
print("Natural logarithm of 10 is:", math.log(10))
print("Logarithm of 100 with base 10 is:", math.log(100, 10))
```

To run the script, open a terminal, navigate to the directory containing `script.py` using the `cd` command (e.g., `cd /path/to/directory`), and type:

```
python3 script.py  # or python script.py on Windows
```

Output:

```
Square root of 16 is: 4.0
Value of pi is: 3.141592653589793
Sine of 90 degrees is: 1.0
Natural logarithm of 10 is: 2.302585092994046
Logarithm of 100 with base 10 is: 2.0
```

Summary

You've now learned Python basics: syntax, variables, control flow, functions, and the math module. Practice all examples instantly at python-fiddle.com.

Chapter 1

Statics

Statics is the branch of mechanics that analyzes bodies in equilibrium under the action of forces. This chapter examines cases where forces are in balance and presents methods for calculating resultant forces and their associated moments.

1.1 Space Diagrams

A space diagram illustrates the geometry of the system and the applied forces.

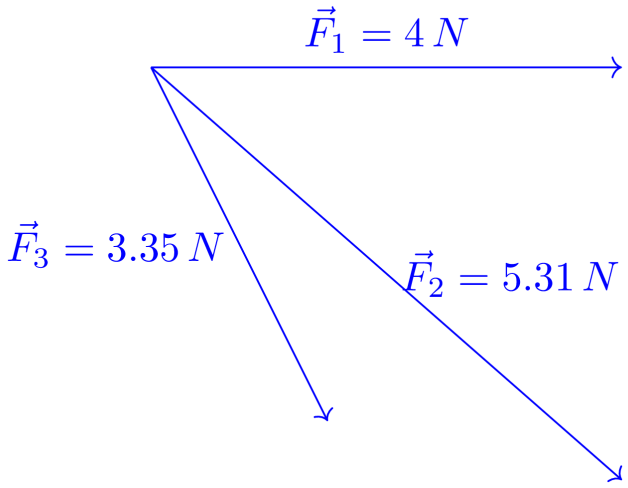


Figure 1.1: Space Diagram

1.2 Vector Diagrams

A vector diagram (or force polygon) is a **scaled** graphical construction in which force vectors are placed head-to-tail in succession. It is used to determine the resultant of a system of forces through vector addition, as well as to resolve equilibrium problems when the polygon closes.

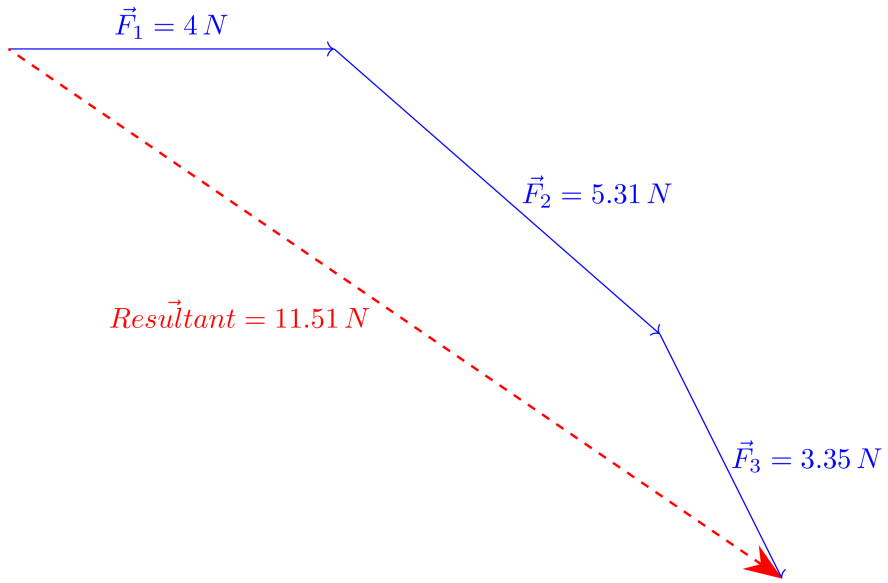


Figure 1.2: Vector Diagram

1.2.1 Resultant

The resultant is a single force that has exactly the same effect on an object as all the original forces acting together. It is found by adding the forces vectorially taking into account both their magnitude and direction. The resultant gives the overall direction and magnitude of the combined forces as shown in Figure 1.2.

1.2.2 Equilibrium

Equilibrium of an object occurs when all the forces acting on it are balanced, so the object remains at rest or moves at a constant speed in a straight line. In equilibrium, as shown in Figure 1.3, there is no net force or acceleration, meaning the object is in a stable state without any change in its motion.

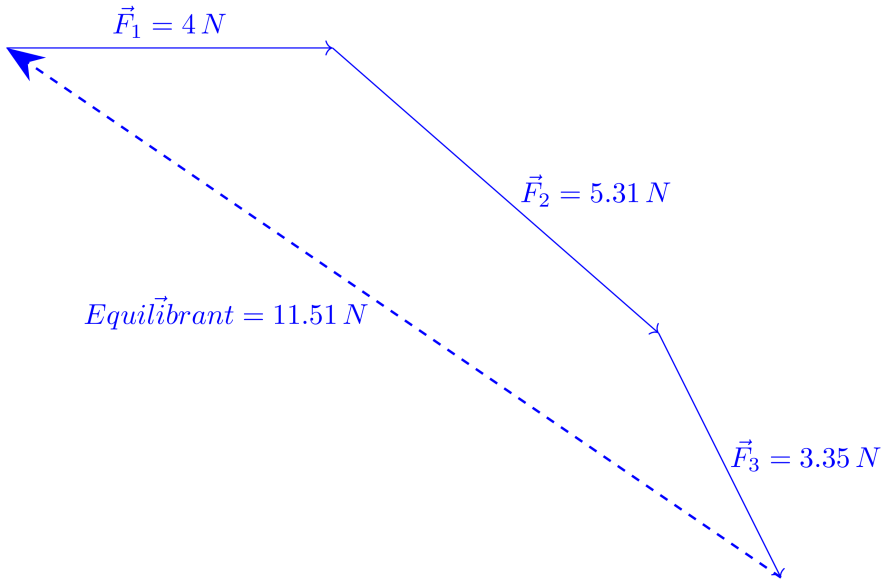


Figure 1.3: Equilibrant

1.3 Conditions of Equilibrium

1. Net force must be zero:

$$\sum_k \vec{F}_k = \vec{0} \tag{1.1}$$

2. Net torque must be zero:

$$\sum_k \vec{\tau}_k = \vec{0} \tag{1.2}$$

1.4 Cosine Rule

The Cosine Rule is used to relate the lengths of the sides of a triangle as shown in Figure 1.4 to the cosine of one of its angles:

$$c^2 = a^2 + b^2 - 2ab\cos\gamma \tag{1.3}$$

Where:

- a, b, c are the sides of the triangle.
- γ is the angle opposite side c.

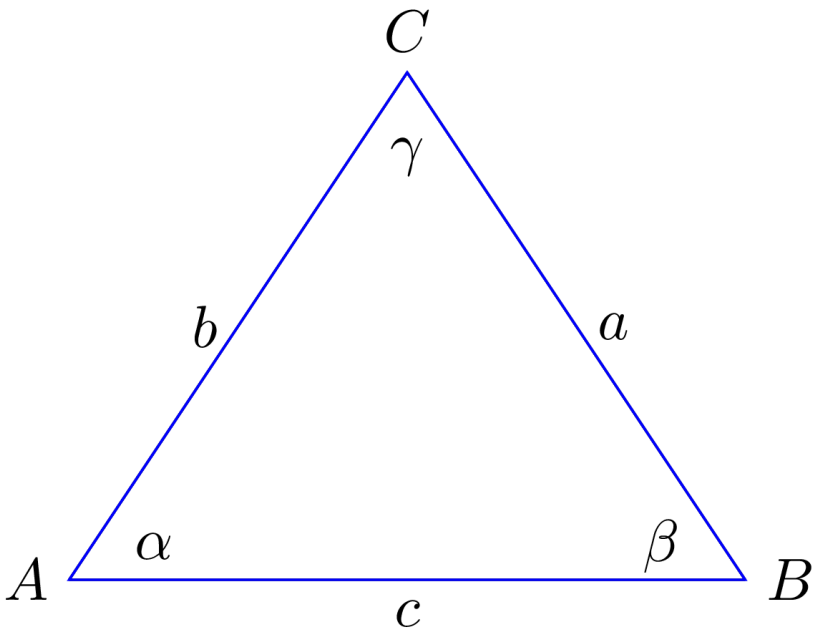


Figure 1.4: Rules of Cosine and Sine

1.5 Sine Rule

The Sine Rule relates the sides and angles of a triangle in Figure 1.4:

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} \tag{1.4}$$

Where:

- α, β, γ are the angles of the triangle.
- a, b, c are the sides of the triangle opposite to angles α, β, γ respectively.

1.6 Problem Set

Example 1.1. Two slings of equal length are slung from a horizontal beam and connected to a ring at their lower ends, the slings and beam forming an equilateral triangle. Find the force in each sling when a load of 30 kN hangs from the ring.

> **Note**
>
> In Python, the math module's trigonometric functions (sin, cos, tan, etc.) expect angles in radians, not degrees. Always convert degrees to radians before using sin, cos, or tan in Python's math module.

```python
import math

# Given
load = 30  # kN
angle_deg = 60  # each sling makes 60° with the horizontal

# Convert angle to radians
angle_rad = math.radians(angle_deg)

# Vertical equilibrium: 2 * T * sin(angle) = load
T = load / (2 * math.sin(angle_rad))

print(f"Tension in each sling: {T:.4f} kN")
```

Example 1.2. Two identical slings of equal length are attached to a horizontal beam and meet at a ring from which a 45 kN load is suspended. The slings and the beam form an isosceles triangle in which each sling makes an angle of 50° with the horizontal. Find the force (tension) in each sling.

```python
import math
```

```
# Given
load = 45      # kN
angle_deg = 50 # degrees

# Convert angle to radians
angle_rad = math.radians(angle_deg)

# Compute tension
T = load / (2 * math.sin(angle_rad))

print(f"Tension in each sling: {T:.4f} kN")
```

Example 1.3. Two lifting ropes are connected at their lower ends to a common shackle from which a load of 25 kN hangs. If the ropes make angles of 32° and 42° respectively to the vertical, find the tension in each rope.

```
import math

# Given
load = 25        # kN
theta1 = 32      # angle opposite T2
theta2 = 42      # angle opposite T1
theta3 = 180 - (theta1 + theta2)  # angle opposite load

# Convert to radians

t1 = math.radians(theta1)
t2 = math.radians(theta2)
t3 = math.radians(theta3)

# Law of sines

T1 = load * math.sin(t2) / math.sin(t3)  # tension in rope 1
T2 = load * math.sin(t1) / math.sin(t3)  # tension in rope 2

print(f"Tension in rope 1: {T1:.4f} kN")
print(f"Tension in rope 2: {T2:.4f} kN")
```

Example 1.4. The angle between the jib and vertical post of a jib crane is 40°, and the angle between the jib and tie is 45°. Find the force in the jib and tie when a load of 15 kN hangs from the crane head.

```
import math

# Given
load = 15  # kN
angle_jib_vertical = 40  # degrees
```

```
angle_jib_tie = 45       # degrees

# Third angle in the force triangle

angle_tie_load = 180 - angle_jib_vertical - angle_jib_tie  # degrees

# Convert angles to radians for math.sin

angle_jib_vertical_rad = math.radians(angle_jib_vertical)
angle_jib_tie_rad = math.radians(angle_jib_tie)
angle_tie_load_rad = math.radians(angle_tie_load)

# Law of sines

F_jib = load * math.sin(angle_tie_load_rad) / math.sin(angle_jib_tie_rad)
F_tie = load * math.sin(angle_jib_vertical_rad) / math.sin(angle_jib_tie_rad)

# Print results
print(f"Force in jib: {F_jib:.4f} kN")
print(f"Force in tie: {F_tie:.4f} kN")
```

Example 1.5. The lengths of the vertical post, jib, and tie of a jib crane are 8, 13, and 9 m, respectively. Find the forces in the jib and tie when a load of 20 kN hangs from the crane head.

```
import math

# Given
P = 8   # post m
J = 13  # jib  m
T = 9   # tie m
W = 20  # load in kN

# Law of cosines

# Angle opposite P (theta_p)
theta_p = math.acos((J**2 + T**2 - P**2) / (2 * J * T))

# Angle opposite T (theta_t)
theta_t = math.acos((J**2 + P**2 - T**2) / (2 * J * P))

# Angle opposite J (theta_j)
theta_j = math.pi - theta_p - theta_t

# Law of sines
```

```
F_J = W * math.sin(theta_j) / math.sin(theta_p)  # force in jib
F_T = W * math.sin(theta_t) / math.sin(theta_p)  # force in tie

# Convert angles to degrees

theta_p_deg = math.degrees(theta_p)
theta_t_deg = math.degrees(theta_t)
theta_j_deg = math.degrees(theta_j)

print(f"theta_p = {theta_p_deg:.4f} degrees")
print(f"theta_t = {theta_t_deg:.4f} degrees")
print(f"theta_j = {theta_j_deg:.4f} degrees")

print()
print(f"Force in jib (F_J): {F_J:.4f} kN")
print(f"Force in tie (F_T): {F_T:.4f} kN")
```

Example 1.6. A ship sailing due east at 18 knots runs into a 3-knot current moving 40° east of north. Find the resultant speed and direction of the ship.

> **Note**
>
> Degrees, Minutes, and Seconds (DMS) notation is widely used in navigation and surveying because decimal degrees alone often lack sufficient precision for small-scale positioning. On Earth's surface:
>
> - 1° of latitude 111 km
> - 1′ (minute of arc) 1.85 km
> - 1″ (second of arc) 30 m
>
> Thus, a position given as 40° 26′ 46″ N instantly conveys location to within tens of metres, whereas the equivalent decimal form, 40.44611°, is less intuitive at a glance despite being mathematically identical.

```
import math

# Given
ship = 18          # knots (due east)
current = 3        # knots
angle_deg = 50     # angle between ship and current in degrees

# Convert to radians

angle_rad = math.radians(angle_deg)
```

```
# Law of cosines

R = math.sqrt(ship**2 + current**2 + 2*ship*current*math.cos(angle_rad))

# Law of sines

theta_rad = math.asin((current * math.sin(angle_rad)) / R)
theta_deg = math.degrees(theta_rad)

# Convert direction to degrees, minutes, seconds

deg = int(theta_deg)
minutes_float = (theta_deg - deg) * 60
minutes = int(minutes_float)
seconds = (minutes_float - minutes) * 60

print(f"Resultant speed: {R:.4f} knots")
print(f"Direction: {theta_deg:.4f}° north of east")
print(f"Direction (DMS): {deg}° {minutes}' {seconds:.0f}\" north of east")
```

Chapter 2

Kinematics

2.1 Linear Motion Definitions

- **Speed**(v): The scalar measure of the rate of change of distance. Mathematically,
$$v = \frac{d}{t}$$
where (d) is distance and (t) is time.
- **Velocity**($\vec{v}$): The vector measure of the rate of change of displacement. It is given by:
$$\vec{v} = \frac{\Delta \vec{x}}{\Delta t}$$
where $\Delta \vec{x}$ is the displacement vector and Δt is the time interval.
- **Acceleration**($\vec{a}$): The rate of change of velocity with respect to time. It is defined as:
$$\vec{a} = \frac{\Delta \vec{v}}{\Delta t}$$

Example 2.1. A ship's engines are stopped when travelling at a speed of 18 knots, and the ship comes to rest after 20 min. Assuming uniform deceleration, find the deceleration (m/s^2) and the distance travelled in nautical miles in that time.

```
import math

# Given

v_knots = 18           # initial speed in knots
t_minutes = 20         # time to stop in minutes
kn_to_ms = 0.5144      # conversion factor (m/s per knot)
```

```
nmi_in_m = 1852        # meters in a nautical mile

# Convert units

vi = v_knots * kn_to_ms      # initial speed in m/s
vf = 0                       # final speed
t = t_minutes * 60           # seconds

# Deceleration (positive magnitude)

deceleration = (vi - vf) / t     # m/s^2

# Average velocity for uniform deceleration

v_avg = (vi + vf) / 2            # m/s

# Distance travelled

s = v_avg * t                    # meters
s_nmi = s / nmi_in_m             # nautical miles

print(f"Deceleration: {deceleration:.6f} m/s^2")
print(f"Distance travelled: {s_nmi:.3f} nautical miles")
```

2.2 Equations of Motion (Constant Acceleration)

$$\vec{v} = \vec{u} + \vec{a}t$$

$$\vec{s} = \frac{\vec{u} + \vec{v}}{2}t$$

$$\vec{s} = \vec{u}t + \frac{1}{2}\vec{a}t^2$$

$$\vec{v}^2 = \vec{u}^2 + 2\vec{a} \cdot \vec{s}$$

where:

- $\vec{u}$: Initial velocity
- $\vec{v}$: Final velocity
- $\vec{s}$: Displacement
- $\vec{a}$: Acceleration

- t: Time

Example 2.2. A ship's propellers are stopped when travelling at 25 knots. From that point, it travels 4 kilometres before coming to a complete stop. Calculate the time it takes to stop in minutes and the average deceleration in m/s^2.

```
# Given

u_knots = 25          # initial speed in knots
v_knots = 0           # final speed in knots
s_m = 4000            # stopping distance in meters
kn_to_ms = 0.5144     # conversion factor knots to m/s

# Convert to m/s

u = u_knots * kn_to_ms
v = v_knots * kn_to_ms

# Average deceleration (from s = 1/2*(u+v)*t)

t = (2 * s_m) / (u + v)

# Deceleration using v = u - a*t

a = (u - v) / t

# Convert time to minutes

t_minutes = t / 60

print(f"Average deceleration: {a:.5f} m/s²")
print(f"Time to stop: {t_minutes:.2f} minutes")
```

Example 2.3. A boat is rowed at 6 km/h perpendicular to a river flowing at 4 km/h. The river is 50 meters wide. How far downstream will the boat reach the opposite bank? What is the boat's actual velocity (magnitude and direction)?

We treat this as vector addition: the boat's rowing velocity across the river and the river's current velocity downstream.

Given

- Boat speed across: $v_b = 6$ km/h
- River current speed: $v_c = 4$ km/h
- River width: $w = 50$ m

Convert speeds to m/s:

$$1 \text{ km/h} = \frac{1000}{3600} = 0.277777 \text{ m/s}$$

$$v_b = 6 \times 0.277777 = 1.6666667 \text{ m/s}$$

$$v_c = 4 \times 0.277777 = 1.1111111 \text{ m/s}$$

Time to cross the river

$$t = \frac{w}{v_b} = \frac{50}{1.6666667} = 30 \text{ s}$$

Downstream drift

$$\text{drift} = v_c \times t = 1.1111111 \times 30 = 33.3333 \text{ m}$$

Actual velocity magnitude

$$v = \sqrt{v_b^2 + v_c^2} = 2.00308 \text{ m/s}$$

Direction (downstream from perpendicular)

$$\theta = \tan^{-1}\left(\frac{v_c}{v_b}\right) = 33.69$$

```python
import math

# Given
boat_speed_kmh = 6      # km/h (perpendicular to river)
current_speed_kmh = 4   # km/h (downstream)
width = 50              # meters (river width)

# Convert speeds to m/s

boat_speed = boat_speed_kmh * 1000 / 3600
current_speed = current_speed_kmh * 1000 / 3600

# Time to cross

t = width / boat_speed

# Downstream drift
```

```
drift = current_speed * t

# Actual velocity (magnitude)

v_actual = math.sqrt(boat_speed**2 + current_speed**2)

# Convert to km/h

v_actual_kmh = v_actual * 3.6

# Direction (downstream from perpendicular)

theta_deg = math.degrees(math.atan(current_speed / boat_speed))

print(f"Time to cross: {t:.2f} s")
print(f"Downstream drift: {drift:.2f} m")
print(f"Actual speed: {v_actual:.2f} m/s ({v_actual_kmh:.2f} km/h)")
print(f"Direction: {theta_deg:.2f} degrees downstream")
```

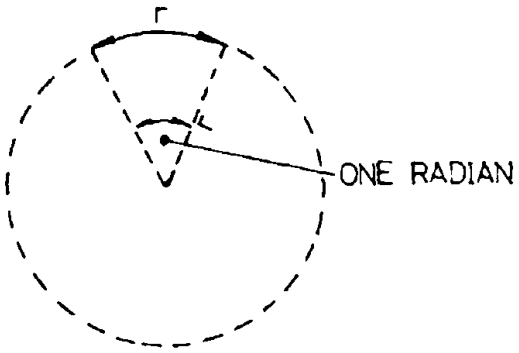
Example 2.4. Two ships are moving toward each other in still water. Ship 1 is 310 meters long and travelling at 22 knots, while Ship 2 is 350 meters long and travelling at 19 knots. Calculate the time it takes for the ships to completely pass each other..

```
import math

# Given

length1 = 310   # meters (Ship 1)
length2 = 350   # meters (Ship 2)
speed1 = 22     # knots
speed2 = 19     # knots

# Total distance to be closed (sum of lengths)

distance_m = length1 + length2  # meters

# Convert meters to nautical miles

distance_nm = distance_m / 1852

# Relative (closing) speed in knots

speed_rel = speed1 + speed2
```

```
# Time in hours

time_hours = distance_nm / speed_rel

# Convert to seconds
time_seconds = time_hours * 3600

print(f"Time for the ships to pass each other: {time_seconds:.2f} seconds")
```

2.3 Angular Motion Definitions

- **Angular Displacement**(θ): The angle through which an object rotates, measured in radians.



$$1 \text{ revolution } = 2\pi \text{ rad}$$

- **Angular Velocity**(ω): The rate of change of angular displacement. Mathematically:

$$\omega = \frac{\Delta\theta}{\Delta t}$$

$$\omega(rad/s) = 2\pi n \; where \; n = speed \; in \; rev/s$$

- **Angular Acceleration**(α): The rate of change of angular velocity with respect to time:

$$\alpha = \frac{\Delta\omega}{\Delta t}$$

2.4 Equations of Angular Motion (Constant Angular Acceleration)

$$\omega_2 = \omega_1 \mp \alpha t$$

$$\theta = \frac{\omega_1 + \omega_2}{2} t$$

$$\theta = \omega_1 t \mp \frac{1}{2} \alpha t^2$$

$$\omega_2^2 = \omega_1^2 \mp 2\alpha\theta$$

where:

- ω_1: Initial angular velocity (rad/s)
- ω_2: Final angular velocity (rad/s)
- θ Angular displacement (rad)
- α: Angular acceleration (rad/s^2)
- t: Time (s)

Example 2.5. A shaft is rotating at 40 revolutions per minute (rev/min). It is uniformly decelerated at a rate of 0.017 rad/s² for 15 seconds. Calculate: 1. The angular velocity of the shaft at the end of 15 seconds (in rad/s and rev/min). 2. The total number of revolutions completed by the shaft during these 15 seconds.

```
import math

# Given

omega1_rpm = 40 # initial angular speed in rev/min
alpha = -0.017  # angular acceleration (rad/s^2), (-) for deceleration
t = 15          # time in seconds

# Convert initial speed to rad/s

omega1 = omega1_rpm * 2 * math.pi / 60

# Angular velocity after time t

omega2 = omega1 + alpha * t

# Angular displacement during time t
```

```
theta = omega1 * t + 0.5 * alpha * t**2

# Convert angular displacement to revolutions

revolutions = theta / (2 * math.pi)

# Convert omega2 to rpm

omega2_rpm = omega2 * 60 / (2 * math.pi)

print(f"Final angular velocity: {omega2:.4f} rad/s")
print(f"Final angular velocity: {omega2_rpm:.4f} rpm")
print(f"Angular displacement: {revolutions:.4f} revolutions")
```

2.5 Relation Between Linear and Angular Motion

The relationship between linear and angular motion is described by the following equations:

$s = r\theta$ (linear displacement s and angular displacement θ).

$v = r\omega$ (linear velocity v and angular velocity ω).

$a = r\alpha$ (linear acceleration a and angular acceleration α).

2.5.1 Variables

- v: Linear velocity, the rate of change of linear displacement ($v = \frac{ds}{dt}$).
- a: Linear acceleration, the rate of change of linear velocity ($a = \frac{dv}{dt}$).
- s: Linear displacement, the distance moved along the circular path.
- r: Radius of the circular path.
- ω: Angular velocity, the rate of change of angular displacement ($\omega = \frac{d\theta}{dt}$).
- α: Angular acceleration, the rate of change of angular velocity ($\alpha = \frac{d\omega}{dt}$).
- θ: Angular displacement, the angle swept by the radius in radians.

2.6 Summary

- Linear motion is directly proportional to angular motion, with the radius (r) acting as the proportionality constant.
- Units for the variables:

- v: meters per second (m/s),
- a: meters per second squared (m/s^2),
- s: meters (m),
- r: meters (m),
- ω: radians per second (rad/s),
- α: radians per second squared (rad/s^2),
- θ: radians (rad).

Chapter 3

Dynamics

Dynamics is the branch of mechanics that studies the motion of bodies under the action of forces. This chapter examines situations where forces produce acceleration, and it presents methods for analyzing the resulting motion, including the application of Newton's laws, energy principles, and momentum conservation.

3.1 Inertia

Inertia is the property of a body that resists any change in its state of motion and is directly proportional to its mass. It can be understood as a form of "sluggishness" inherent to matter. For an object at rest, a net force is required to set it in motion; the greater the mass, the greater the force needed. Similarly, if an object is already moving, a net force is required to change its speed or direction, and the magnitude of this force is again proportional to the mass.

The term "inertia" is also used in structural mechanics to describe the resistance of a beam's cross-section to bending, quantified by the second moment of area (see lecture notes on Second Moments).

3.2 Momentum

Momentum is the product of an object's mass and its velocity and quantifies the amount of motion a moving body possesses. This concept is particularly important when considering a large vessel approaching a dock. Once the vessel is underway, its considerable momentum means that a substantial force, applied over sufficient time, is required to bring it to a stop or significantly alter its course. Without such intervention, the vessel will not stop quickly on its own and may collide with the dock.

3.3 Newton's Laws of Motion

1. **First Law (Law of Inertia)**:
   An object remains at rest, or in uniform motion in a straight line, unless acted upon by a net external force.

$$\Sigma F = 0 \implies v = \text{constant}$$

2. **Second Law (Law of Acceleration)**:
   The acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass.

$$\vec{F} = m\vec{a}$$

3. **Third Law (Action-Reaction Law)**:
   For every action, there is an equal and opposite reaction.

$$\vec{F}_{\text{action}} = -\vec{F}_{\text{reaction}}$$

Example 3.1. Find the accelerating force required to increase the velocity of a body which has a mass of 20 kg from 30 m/s to 70 m/s in 4 s.

```
# Given

m = 20           # mass in kg
u = 30           # initial velocity in m/s
v = 70           # final velocity in m/s
t = 4            # time in seconds

# Calculate acceleration

a = (v - u) / t

# Calculate force

F = m * a

print(f"Acceleration: {a:.2f} m/s²")
print(f"Accelerating force: {F:.2f} N")
```

Example 3.2. A lift is supported by a steel wire rope, the total mass of the lift and contents is 750 kg. Find the tension in the wire rope, in newtons, when the lift is (i) moving at constant velocity, (ii) moving upwards and accelerating at 1.2 m/s^2, (iii) moving upwards and retarding at 1.2 m/s^2

```
# Given
```

```
m = 750          # mass in kg
g = 9.81         # gravitational acceleration in m/s^2

# Accelerations for each case

a_constant = 0
a_up_accel = 1.2
a_up_decel = -1.2

# Tension calculations

T_constant = m * (g + a_constant)
T_up_accel = m * (g + a_up_accel)
T_up_decel = m * (g + a_up_decel)

print(f"Tension at constant velocity: {T_constant:.4f} N")
print(f"Tension during upward acceleration: {T_up_accel:.4f} N")
print(f"Tension during upward deceleration: {T_up_decel:.4f} N")
```

3.4 Linear momentum

Linear momentum is a fundamental concept in physics that quantifies the motion of an object. It is defined as the product of the object's mass and its velocity. Because momentum is a vector quantity, it possesses both magnitude and direction.

$$Linear\ momentum = m\vec{v}$$

Where:

- Linear momentum is in $\text{kg} \cdot \text{m/s}$.
- m is the mass of the object in kilograms.
- $\vec{v}$ is the velocity of the object in meters per second.

3.4.1 Conservation of Linear Momentum

The law of conservation of linear momentum states that, in a closed system with no external forces (or when the net external force is zero), the total linear momentum remains constant over time. This principle is particularly evident in collisions between two bodies. During a collision, the force exerted by the first body on the second is equal in magnitude and opposite in direction to the force exerted by the second on the first (Newton's third law). These equal and opposite forces act for the same duration, producing changes in momentum that

are equal in magnitude but opposite in direction. Consequently, the momentum gained by one body exactly equals the momentum lost by the other. Therefore, the total momentum of the system before the collision is equal to the total momentum after the collision.

In the absence of external forces, linear momentum is neither created nor destroyed; it is conserved. This is known as the law of conservation of linear momentum.

Example 3.3. A hand hammer with a head mass of 0.8 kg strikes a chisel at 9 m/s and comes to rest in 1/250 seconds. Calculate the average force exerted during the blow.

```
# Given

m = 0.8               # kg
u = 9                 # m/s (initial velocity)
v = 0                 # m/s (final velocity)
t = 1/250             # s (time to come to rest)

# Change in momentum

delta_p = m * (v - u)

# Average force

F_avg = delta_p / t

print(f"Average force during the blow: {F_avg:.4f} N")
```

Example 3.4. A jet of fresh water, 20 mm in diameter, emerges horizontally from a nozzle at a speed of 21 m/s and strikes a stationary vertical plate normally. Assuming no splashback (i.e., the water comes to rest upon impact with no velocity component normal to the plate after striking), calculate: (a) the mass flow rate of water leaving the nozzle (in kg/s), and (b) the force exerted by the jet on the plate. The density of fresh water is 1000 kg/m^3.

```
import math

# Given

d = 0.02          # diameter in meters
v = 21            # jet speed in m/s
rho = 1000        # density of water in kg/m^3

# Mass flow rate

A = math.pi * (d**2) / 4        # cross-sectional area
```

```
v_dot = A * v                    # volumetric flow rate
m_dot = rho * v_dot              # mass flow rate

# Force on the plate

F = m_dot * v                    # momentum change

print(f"Mass flow rate (kg/s): {m_dot:.4f}")
print(f"Force on plate (N): {F:.4f}")
```

3.5 Angular Momentum

Angular momentum is the moment of linear momentum about a chosen point or axis. For a single particle, it is given by the cross product of its position vector (measured from that point) and its linear momentum.

More broadly, angular momentum measures the rotational motion of a system. For extended bodies, its evaluation depends on both the distance of mass elements from the axis (the first moment of position) and how the mass is distributed throughout the object—the moment of inertia, or second moment of mass.

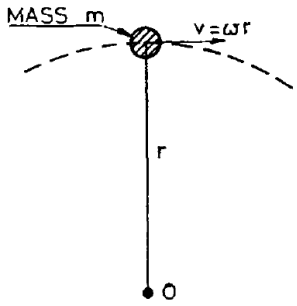


Figure 3.1: Angular momentum

Given **linear velocity** $\vec{v}$ and **angular velocity** ω:

$$\vec{v} = r\omega$$

The **linear momentum** is determined by:

$$\text{Linear momentum} = m\vec{v}$$

Substituting for $\vec{v}$:

$$\text{Linear momentum} = mr\omega$$

The **moment of linear momentum** can then be written as:

$$\text{Moment of linear momentum} = mr\omega r$$

The moment of linear momentum is referred to as the **angular momentum**. Simplifying the above equation:

$$\text{Angular momentum} = mr^2\omega$$

Here, mr^2 is the **moment of inertia** (second moment of mass) of the object about its axis of rotation, denoted as I. Thus, the angular momentum can be expressed as:

$$\text{Angular momentum} = I\omega$$

Where:

- I is the moment of inertia in $kg \cdot m^2$.
- ω is the angular velocity in rad/s.

Additionally, the moment of inertia I is given by:

$$I = mk^2$$

Where:

- I is the moment of inertia in $kg \cdot m^2$.
- m is the mass in kg.
- k is the radius of gyration in m.

> **Note**
>
> In structural engineering, the moment of inertia (m) is a measure of resistance to bending (deflection or stiffness).
> In angular motion and physics, the moment of inertia ($kg \cdot m^2$) is a measure of resistance to angular acceleration (rotational inertia).

3.6 The Radius of Gyration

The radius of gyration is a geometric property of a rigid body or mass distribution that describes how the mass is distributed relative to an axis of rotation. It represents the effective distance from the axis at which the entire mass could be concentrated while producing the same moment of inertia as the actual distributed mass.

In simpler terms, a smaller radius of gyration means the mass is closer to the axis, which is good for reducing inertia in rotating machinery. This is desirable in applications like engine crankshafts or turbine rotors where rapid changes in speed are required. A larger radius of gyration means the mass is farther from the axis, like a hollow cylinder has a larger k than a solid cylinder of the same mass and outer radius. This is beneficial when energy storage is needed (e.g., flywheels).

Example 3.5. A flywheel of mass 500 kg and radius of gyration 1.2 m is running at 300 rev/min. By means of a clutch, this flywheel is suddenly connected to another flywheel, mass 2000 kg and radius of gyration 0.6 m, initially at rest. Calculate their common speed of rotation after engagement.

```
import math

# Given data

m1, k1 = 500, 1.2
m2, k2 = 2000, 0.6
rpm1 = 300

# Moments of inertia

I1 = m1 * k1**2
I2 = m2 * k2**2

# Convert rpm to rad/s

omega1 = (rpm1 / 60) * 2 * math.pi

# Conservation of angular momentum

omega_f = (I1 * omega1) / (I1 + I2)

# Convert rad/s to rpm

rpm_f = (omega_f / (2 * math.pi)) * 60

print(f"Final angular velocity (rad/s): {omega_f:.4f}")
```

```
print(f"Final angular velocity (rpm): {rpm_f:.4f}")
```

3.7 Turning Moment

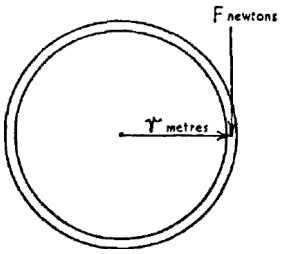


Figure 3.2: Turning moment

We know the relation between linear acceleration a and angular acceleration α:

$$a = r\alpha$$

And

$$F = ma$$

Therefore,

$$F = m\alpha r$$

If the force is not applied directly on the rim but at a greater or lesser leverage, say L from the centre, the effective force on the rim causing it to accelerate will be greater or lesser accordingly, in the ratio of L to r. Thus:

$$F\frac{L}{r} = m\alpha r$$

Multiplying both sides by r,

$$FL = m\alpha r^2$$

Now, FL is the torque applied. Therefore, the accelerating torque is:

$$\tau = mr^2\alpha$$

Or,

$$\tau = mk^2\alpha$$

τ is also given by:

$$\tau = I\alpha$$

Where:

- τ is the torque in Nm.
- I is the moment of inertia in $kg \cdot m^2$.
- α: Angular acceleration in rad/s^2.

Example 3.6. The mass of a flywheel is 175 kg and its radius of gyration is 380 mm. Find the torque required to attain a speed of 500 rev/min from rest in 30 s.

```
import math

# Given
m = 175                 # mass in kg
k = 0.38                # radius of gyration in m
rpm_final = 500         # final speed
t = 30                  # time in seconds

# Moment of inertia
I = m * k**2

# Convert rpm → rad/s
omega_final = (rpm_final * 2 * math.pi) / 60

# Angular acceleration
alpha = omega_final / t

# Torque
tau = I * alpha

print(f"Moment of inertia I (kg·m^2): {I:.4f}")
print(f"Final angular velocity  (rad/s): {omega_final:.4f}")
print(f"Angular acceleration  (rad/s^2): {alpha:.4f}")
print(f"Required torque  (N·m): {tau:.4f}")
```

Example 3.7. The torque to overcome frictional and other resistances of a turbine is 317 N m and may be considered constant for all speeds. The mass of the rotating parts is 1.59 t and the radius of gyration is 0.686 m. If the gas is

cut off when the turbine is running free of load at 1920 rev/min, find the time it will take to come to rest and the number of revolutions turned during that time.

```
import math

# Given

tau = 317.0           # N·m
m = 1.59 * 1000       # kg
k = 0.686             # m
rpm0 = 1920.0         # rev/min


I = m * k**2
omega0 = (rpm0 / 60.0) * 2.0 * math.pi   # rad/s
alpha = tau / I                          # rad/s^2
t = omega0 / alpha                       # s
theta = 0.5 * omega0 * t                 # rad
revs = theta / (2.0 * math.pi)


print(f"I (kg·m^2):           {I:.4f}")
print(f"omega0 (rad/s):       {omega0:.4f}")
print(f"alpha (rad/s^2):      {alpha:.6f}")
print(f"time to stop (s):     {t:.4f}")
print(f"time to stop (min):   {t/60:.4f}")
print(f"revolutions:          {revs:.4f}")
```

3.8 Power by Torque

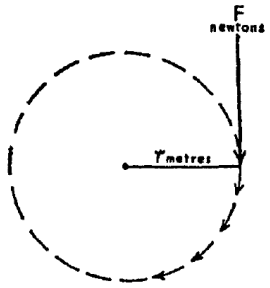


Figure 3.3: A rotating mechanism.

Consider a force F applied at a radius r on a rotating mechanism, as shown above.

The work done in one revolution is the product of the force and the circumference. Therefore:

$$W = F \cdot 2\pi r$$

If the mechanism is running at n revolutions per second:

$$\text{Power} = F \cdot 2\pi r n$$

Since torque $\tau = Fr$, we can rewrite the equation as:

$$P = \tau \cdot 2\pi n$$

Given that $\omega = 2\pi n$:

$$P = \tau \cdot \omega$$

Where:

- P is the power in watts (W),
- τ is the torque in newton-meters (Nm), and
- ω is the angular velocity in radians per second (rad/s).

Example 3.8. The mean torque in a propeller shaft is 2.26 x 10^5 Nm when running at 140 rpm. Find the power transmitted.

```
import math

# Given

T = 2.26e5      # Nm
rpm = 140

omega = rpm * 2 * math.pi / 60
P = T * omega

print(f"Angular speed (rad/s): {omega:.4f}")
print(f"Power (W): {P:.4f}")
print(f"Power (MW): {P/1e6:.4f}")
```

Example 3.9. One gear wheel with 100 teeth of 6 mm pitch running at 250 rev/min drives another which has 50 teeth. If the power transmitted is 0.5 kW, find the driving force on the teeth and the speed of the driven wheel.

```python
import math

# Given

N1 = 100          # teeth on driving wheel
N2 = 50           # teeth on driven wheel
pitch = 0.006     # circular pitch (m)
n1 = 250          # driving rpm
P = 0.5e3         # power in watts

# Pitch diameter of driving wheel

D1 = (N1 * pitch) / math.pi
r1 = D1 / 2

# Angular speed of driving wheel

omega1 = n1 * 2 * math.pi / 60

# Torque on driving wheel

T = P / omega1

# Tangential driving force

F = T / r1

# Driven wheel speed

n2 = n1 * (N1 / N2)

print(f"Pitch diameter D1 (m): {D1:.4f}")
print(f"Torque T (N·m): {T:.4f}")
print(f"Tangential force F (N): {F:.4f}")
print(f"Driven wheel speed n2 (rpm): {n2:.4f}")
```

3.9 Kinetic Energy of Rotation

We know that $K.E. = \frac{1}{2}mv^2$, where v is the linear velocity of the body. For a rotating body, the effective linear velocity is at the radius of gyration, as this is the radius at which the entire mass of the rotating body can be considered to act.

Let k = radius of gyration (in meters),
Let ω = angular velocity (in radians per second).

The relationship between linear and angular velocity is:

$$v = \omega k$$

Substituting $v^2 = \omega^2 k^2$ into $K.E. = \frac{1}{2}mv^2$ gives:

$$\text{Rotational K.E.} = \frac{1}{2}mk^2\omega^2$$

Since $I = mk^2$, where I is the moment of inertia:

$$\text{Rotational K.E.} = \frac{1}{2}I\omega^2$$

Example 3.10. The radius of gyration of the flywheel of a shearing machine is 0.46 m and its mass is 750 kg. Find the kinetic energy stored in it when running at 120 rev/min. If the speed falls to 100 rev/min during the cutting stroke, find the kinetic energy given out by the wheel.

```
import math

m = 750
k = 0.46
I = m * k**2

omega1 = 120 * 2 * math.pi / 60
omega2 = 100 * 2 * math.pi / 60

K1 = 0.5 * I * omega1**2
K2 = 0.5 * I * omega2**2
dK = K1 - K2

print(f"I (kg·m^2):           {I:.4f}")
print(f"KE at 120 rpm:        {K1/1e3:.4f} kJ")
print(f"KE at 100 rpm:        {K2/1e3:.4f} kJ")
print(f"Energy given out:     {dK/1e3:.4f} kJ")
```

Example 3.11. A ship's anchor has a mass of 5 tonnes. Determine the work done in raising the anchor from a depth of 100 m. If the hauling gear is driven by a motor whose output is 80 kW and the efficiency of the haulage is 75%, determine how long the lifting operation takes.

```
# Given

m_anchor = 5 * 1000        # tonnes to kg
depth = 100                # m
```

```
g = 9.81                    # m/s^2

motor_power = 80_000        # W
efficiency = 0.75           # 75%

# Work done lifting the anchor

work_done = m_anchor * g * depth

# Effective power available for lifting

useful_power = motor_power * efficiency

# Time required

time_seconds = work_done / useful_power

print("Work done in lifting the anchor (MJ): {:.4f}".format(work_done/1e6))
print("Time required (s): {:.4f}".format(time_seconds))
```

Chapter 4

Summary of Key Terms

4.0.1 Inertia

The tendency of an object to resist any change in its motion. An object with greater mass has greater inertia (it is "harder to start or stop").

- In **Naval Architecture**, the *moment of inertia* (units: m) is a measure of resistance to bending (related to stiffness/deflection of beams and hull girders).
- In **angular motion and physics**, the *moment of inertia* (units: $\mathrm{kg} \cdot \mathrm{m}^2$) is a measure of resistance to angular acceleration (rotational inertia).

4.0.2 Linear Momentum (p)

Momentum is the "quantity of motion" an object has. It is calculated as:

$$p = mv$$

where
- p = linear momentum ($\mathrm{kg} \cdot \mathrm{m/s}$)
- m = mass (kg)
- v = velocity (m/s)

Momentum is a **vector quantity** (it has both magnitude and direction).

4.0.3 Newton's Laws of Motion

4.0.3.1 First Law (Law of Inertia)

An object will stay at rest or keep moving in a straight line at constant speed unless an unbalanced (net) external force acts on it.

4.0.3.2 Second Law (Law of Acceleration)

The acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass.

$$F_{\text{net}} = ma \quad \text{or} \quad a = \frac{F_{\text{net}}}{m}$$

4.0.3.3 Third Law (Action-Reaction)

Whenever two objects interact, they exert equal and opposite forces on each other.
"For every action force, there is an equal and opposite reaction force."

4.0.4 Conservation of Linear Momentum

If no external forces act on a system (or if the external forces cancel out), the total momentum of the system stays constant.

In collisions or explosions:

Total momentum before = Total momentum after

Example: When two objects collide and stick together or bounce apart, the momentum lost by one is exactly gained by the other, so the total remains the same.

4.0.5 Angular Momentum

The rotational equivalent of linear momentum. It describes how much "rotational motion" a spinning or orbiting object has.

Angular momentum depends on mass, speed, and distance from the axis of rotation.

4.0.6 Radius of Gyration (k)

A geometric property of a rigid body that indicates how the mass is distributed relative to a specified axis of rotation. It is defined such that:

$$I = mk^2$$

where
- I = moment of inertia ($\text{kg} \cdot \text{m}^2$)
- m = total mass (kg)
- k = radius of gyration (m)

4.0.6.1 Practical Examples for Radius of Gyration

- **Smaller** $k \rightarrow$ mass is concentrated closer to the axis $\rightarrow$ **lower** rotational inertia
  $\rightarrow$ easier/faster to speed up or slow down rotation
  (e.g., engine crankshafts, turbine rotors, figure skater pulling arms in).
- **Larger** $k \rightarrow$ mass is distributed farther from the axis $\rightarrow$ **higher** rotational inertia
  $\rightarrow$ better for storing rotational energy
  (e.g., flywheels; a hollow cylinder has a larger k than a solid cylinder of the same mass and outer radius).

Chapter 5

Hydrostatics

Hydrostatics is the branch of fluid mechanics that studies fluids at rest and the forces and pressures they exert on immersed or containing surfaces. Key principles include Pascal's law (pressure applied to an enclosed fluid is transmitted undiminished in all directions), Archimedes' principle (buoyant force on a submerged body equals the weight of displaced fluid), and the hydrostatic pressure distribution in a fluid column, expressed as $p = \rho g h$, where p is gauge pressure, ρ is fluid density, g is gravitational acceleration, and h is depth below the free surface.

5.1 Pressure Head

In fluid mechanics, pressure head is the height of a fluid column that corresponds to a specific pressure at a given point, assuming the fluid is static. It's expressed as $h_p = \frac{p}{\rho g}$, where p is the pressure above atmospheric, ρ is the fluid density, and g is the acceleration due to gravity. The unit is typically meters (or feet) of the fluid column. Pressure head represents the potential energy per unit weight due to pressure and is a component of the total hydraulic head, which is the sum of pressure head and elevation head.

5.2 Load On Immersed Surfaces

The load on an immersed surface is the total hydrostatic force exerted by a fluid on a surface submerged in a static fluid. This force arises from the pressure variation with depth, governed by the hydrostatic pressure equation $p = \rho g h$, where ρ is fluid density, g is gravitational acceleration, and h is the depth below the free surface.

The pressure acts perpendicular to the surface, resulting in a distributed load.

The total force F is calculated as $F = \rho g A \bar{h}$, where A is the surface area and $\bar{h}$ is the depth of the centroid of the area.

Example 5.1. A vertical rectangular bulkhead is 7 m wide and extends over the full height of the water column. Fresh water stands 6 m deep on one side and is assumed to be at zero level on the other side. Calculate the total hydrostatic load (thrust) on the bulkhead. Density of fresh water = 1000 kg/m³.

```
# Given
width          = 7.0    # m (horizontal width of bulkhead)
height_water   = 6.0    # m (water depth on one side)
rho_water      = 1000.0 # kg/m³ (fresh water)
g              = 9.81   # m/s²

# Calculations
area           = width * height_water                    # m²
h_centroid     = height_water / 2                        # m (from free surface)
force_N        = rho_water * g * h_centroid * area       # Newtons
force_kN       = force_N / 1000


print("=== Hydrostatic Load on Bulkhead ===")
print(f"Bulkhead width         : {width} m")
print(f"Water depth            : {height_water} m")
print(f"Submerged area         : {area:.1f} m²")
print(f"Depth of centroid      : {h_centroid} m")
print(f"Total water load       : {force_kN:,.0f} kN  ({force_kN/1000:.3f} MN)")
```

Example 5.2. A 10-meter-long, 4-meter-wide, and 6-meter-high tank is filled with oil (specific gravity = 0.9). The oil rises 5 meters up a vent pipe above the tank's top. Calculate the load on one end plate and the bottom of the tank.

```
# ------------------------------
# Given data
# ------------------------------
tank_length = 10.0  # m
tank_width = 4.0  # m
tank_height = 6.0  # m
sounding_pipe = 5.0  # m (oil rises 5 m above the tank top in vent/pipe)

specific_gravity = 0.9
rho_oil = specific_gravity * 1000  # kg/m³
g = 9.81  # m/s²

# ------------------------------
# Calculated values
# ------------------------------
```

```
total_head = tank_height + sounding_pipe  # from bottom to oil surface

# End plate dimensions and force
end_height = tank_height  # m
end_width = tank_width  # m
area_end = end_height * end_width  # m²

# Depth of centroid of end plate below oil surface
h_centroid_end = sounding_pipe + (tank_height / 2)  # m

# Hydrostatic force on end plate
force_end_plate = rho_oil * g * h_centroid_end * area_end

# Bottom plate force
area_bottom = tank_length * tank_width  # m²
force_bottom = rho_oil * g * total_head * area_bottom

# ------------------------------
# Output results
# ------------------------------
print("=== Hydrostatic Loads on Oil Tank ===")
print(
    f"Tank dimensions       : {tank_length} m × {tank_width} m × "
    f"{tank_height} m (L×W×H)"
)
print(f"Oil rise in sounding pipe : {sounding_pipe} m")
print(f"Oil density           : {rho_oil} kg/m³")
print()
print(
    f"End plate (one wall)  : {end_height} m high × {end_width} m "
    f"wide = {area_end} m²"
)
print(
    f"Horizontal load on one end plate : {force_end_plate:,.0f} N = "
    f"{force_end_plate/1000:.0f} kN"
)
print()
print(
    f"Bottom plate          : {tank_length} m × {tank_width} m = "
    f"{area_bottom} m²"
)
print(
    f"Vertical load on bottom          : {force_bottom:,.0f} N = "
    f"{force_bottom/1000:.0f} kN"
)
```

Example 5.3. A rectangular dock gate measures 4 meters wide. If the water level is 5.5 meters high on one side and 3.5 meters high on the other, what horizontal thrust will act on the gate? Assume the water density is 1000 kg/m³.

```
# Given

width = 4.0          # gate width in meters
h1 = 5.5             # water depth on higher side (m
h2 = 3.5             # water depth on lower side (m)
rho = 1000.0         # density of water (kg/m³)
g = 9.81             # acceleration due to gravity (m/s²)

# Hydrostatic force on higher side
A1 = h1 * width
h_c1 = h1 / 2
F1 = rho * g * h_c1 * A1

# Hydrostatic force on lower side
A2 = h2 * width
h_c2 = h2 / 2
F2 = rho * g * h_c2 * A2

# Net horizontal thrust (from higher to lower side)
F_net = F1 - F2

print("=== Hydrostatic Thrust Calculation ===")
print(f"Force from higher side (5.5 m): {F1:.0f} N")
print(f"Force from lower side  (3.5 m): {F2:.0f} N")
print(f"Net horizontal thrust on gate: {F_net:.0f} N")
```

Example 5.4. A rectangular dock gate measures 12 meters wide. When the sea is 9 meters deep on one side and 4.5 meters deep on the other, what horizontal thrust will act on the gate? Assume the density of seawater is 1025 kg/m³.

```
# ------------------------------
# Given data
# ------------------------------
width = 12.0         # width of gate (m)
h1 = 9.0             # deeper side water depth (m)
h2 = 4.5             # shallower side water depth (m)
rho = 1025.0         # density of seawater (kg/m³)
g = 9.81             # acceleration due to gravity (m/s²)

# ------------------------------
# Hydrostatic force on deeper side
# ------------------------------
```

```
A1 = h1 * width           # area of gate submerged
h_c1 = h1 / 2             # depth of centroid
F1 = rho * g * h_c1 * A1  # hydrostatic force

# ----------------------------
# Hydrostatic force on shallower side
# ----------------------------
A2 = h2 * width           # area of gate submerged
h_c2 = h2 / 2             # depth of centroid
F2 = rho * g * h_c2 * A2  # hydrostatic force

# ----------------------------
# Net horizontal thrust
# ----------------------------
F_net = F1 - F2

# ----------------------------
# Output results
# ----------------------------
print("=== Dock Gate Hydrostatic Thrust Calculation ===")
print(f"Force from {h1} m side : {F1:,.0f} N  ({F1/1e6:.3f} MN)")
print(f"Force from {h2} m side : {F2:,.0f} N  ({F2/1e6:.3f} MN)")
print(
    f"Resultant horizontal thrust: {F_net:,.0f} N  "
    f"({F_net/1000:.1f} kN or {F_net/1e6:.3f} MN)"
)
```

Example 5.5. A vertical rectangular bulkhead, 2 meters wide, divides two liquids in a tank. On one side, oil with a density of 850 kg/m^3 stands 4 meters deep. On the other side, fresh water with a density of 1000 kg/m^3 is 6 meters deep. The bottom of the bulkhead rests on the tank bottom, and both liquids are open to the atmosphere at the top, resulting in zero gauge pressure at the free surfaces. Calculate the magnitude and direction of the net hydrostatic thrust on the bulkhead.

```
# ----------------------------
# Given data
# ----------------------------
width = 2.0  # m (width of bulkhead)
h_oil = 4.0  # m (oil depth)
h_water = 6.0  # m (water depth)
rho_oil = 850.0  # kg/m³
rho_water = 1000.0  # kg/m³
g = 9.81  # m/s²

# ----------------------------
```

```
# Display input parameters
# ------------------------------
print("Bulkhead Separating Oil and Water")
print(f"Bulkhead width          : {width} m")
print(f"Oil side    : {h_oil} m deep,     = {rho_oil} kg/m³")
print(f"Water side  : {h_water} m deep,     = {rho_water} kg/m³")
print()

# ------------------------------
# Hydrostatic force on oil side
# ------------------------------
A_oil = width * h_oil
h_c_oil = h_oil / 2
F_oil = rho_oil * g * h_c_oil * A_oil

# ------------------------------
# Hydrostatic force on water side
# ------------------------------
A_water = width * h_water
h_c_water = h_water / 2
F_water = rho_water * g * h_c_water * A_water

# ------------------------------
# Net horizontal thrust
# ------------------------------
F_net_N = F_water - F_oil  # positive towards oil side
F_net_kN = F_net_N / 1000

# ------------------------------
# Output results
# ------------------------------
print(f"Force from oil side     : {F_oil:8.0f} N  = {F_oil/1000:6.1f} kN")
print(f"Force from water side   : {F_water:8.0f} N  = {F_water/1000:6.1f} kN")
print()
print(f"NET THRUST              : {F_net_N:8.0f} N  = {F_net_kN:6.1f} kN")
print("Direction               : Towards the oil side")
```

Example 5.6. Seawater has flooded an oil tanker's oil tank to a depth of 4 meters, and a 10-meter layer of oil sits atop the seawater. Calculate the pressure at the bottom of the tank. Note: The density of oil is 0.85 g/cm³, the density of seawater is 1.02 t/m³, and the atmospheric pressure is 101.3 kPa.

```
# ------------------------------
# Given data
# ------------------------------
oil_density_g_per_cm3 = 0.85  # Density of oil in g/cm³
```

```
oil_height_m = 10.0  # Height of oil layer in meters
seawater_density_t_per_m3 = 1.02  # Density of seawater in t/m³
seawater_height_m = 4.0  # Height of seawater layer in meters
atmospheric_pressure_kPa = 101.3  # Atmospheric pressure in kPa
g = 9.81  # Acceleration due to gravity (m/s²)

# ------------------------------
# Convert densities to kg/m³
# ------------------------------
oil_density = oil_density_g_per_cm3 * 1000  # g/cm³ → kg/m³
seawater_density = seawater_density_t_per_m3 * 1000  # t/m³ → kg/m³

# ------------------------------
# Hydrostatic pressure calculations
# ------------------------------
pressure_oil_kPa = (oil_density * g * oil_height_m) / 1000
pressure_seawater_kPa = (seawater_density * g * seawater_height_m) / 1000

# Total gauge and absolute pressure
gauge_pressure_kPa = pressure_oil_kPa + pressure_seawater_kPa
absolute_pressure_kPa = atmospheric_pressure_kPa + gauge_pressure_kPa

# ------------------------------
# Output results
# ------------------------------
print(f"Pressure due to oil layer       : {pressure_oil_kPa:.4f} kPa")
print(f"Pressure due to seawater layer  : {pressure_seawater_kPa:.4f} kPa")
print(f"Total gauge pressure            : {gauge_pressure_kPa:.4f} kPa")
print(f"Absolute pressure at the bottom : {absolute_pressure_kPa:.4f} kPa")
```

5.3 Hydraulic Jacks

Hydraulic jacks use Pascal's law to lift heavy loads with minimal input force. Pascal's law states that pressure applied to an enclosed fluid is transmitted equally in all directions. In a hydraulic jack, two pistons of different sizes are connected by a confined fluid. Applying an effort force to the smaller piston generates a pressure that is transmitted to the larger piston, producing an output force. Calculations involving hydraulic jacks, such as determining system pressure, required input force, or lifted load, rely on this principle under ideal conditions, neglecting friction and fluid compressibility.

Example 5.7. A force of 150 N is transmitted from a piston of 25 mm² to one of 100 mm². Determine the system pressure and the load carried by the larger piston.

```
# ------------------------------
# Given parameters
# ------------------------------
area_small_mm2 = 25.0  # mm² (small piston)
area_large_mm2 = 100.0  # mm² (large piston)
force_small = 150.0  # N (force applied to small piston)

# ------------------------------
# Convert areas to SI units (m²)
# ------------------------------
area_small = area_small_mm2 * 1e-6  # 1 mm² = 10^-6 m²
area_large = area_large_mm2 * 1e-6

# ------------------------------
# System pressure and load on large piston
# ------------------------------
pressure = force_small / area_small  # Pa
load_large = pressure * area_large  # N

# ------------------------------
# Output results
# ------------------------------
print(f"System pressure: {pressure:.2f} Pa " f"(or {pressure / 1e6:.1f} MPa)")
print(f"Load carried by larger piston: {load_large:.1f} N")
```

Example 5.8. A simple hydraulic jack consists of a small effort plunger with a diameter of 25 mm and a large load piston with a diameter of 70 mm. The jack lifts a load of 5.88 kN. Calculate:

1. The pressure in the hydraulic fluid (system pressure).
2. The force exerted on the small plunger (load on the small plunger) required to support the given load, assuming ideal conditions (no losses).

```
import math

# ------------------------------
# Given data
# ------------------------------
diameter_small = 0.025  # m (effort plunger)
diameter_large = 0.070  # m (load piston)
load = 5880.0  # N (equivalent to 5.88 kN)

# ------------------------------
# Calculate radii and cross-sectional areas
# ------------------------------
radius_small = diameter_small / 2
```

```
radius_large = diameter_large / 2

area_small = math.pi * (radius_small**2)  # m²
area_large = math.pi * (radius_large**2)  # m²

# ------------------------------
# System pressure and ideal force on small plunger
# ------------------------------
pressure = load / area_large  # Pa
force_small = pressure * area_small  # N

# ------------------------------
# Output results
# ------------------------------
print(f"System pressure: {pressure:.2f} Pa " f"(or {pressure / 1e6:.3f} MPa)")
print(f"Ideal force on small plunger: {force_small:.1f} N")
```

Chapter 6

Hydrodynamics

Hydrodynamics, a branch of fluid mechanics, studies the motion of liquids and the forces acting on them. Unlike aerodynamics, which concerns compressible gases such as air, hydrodynamics primarily deals with incompressible fluids like water. It is based on classical physics principles, including Newton's laws of motion, the conservation of mass and momentum. These principles are used to describe the velocity, pressure and viscosity of a fluid in motion.

6.1 Volume Flow Rate

Volume rate of flow refers to the rate at which a fluid volume passes a given section in a flow stream. Volume rate of flow may also be referred to as capacity of flow, flow rate or discharge. It is generally given in units of volume per unit of time, cubic meters per second (m^3/s) or liters per second (L/s).

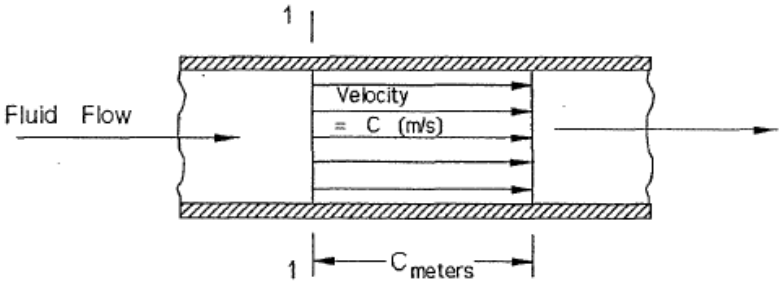


Figure 6.1: Volume flow

Consider an ideal fluid flow in a pipe of cross-section 'A' (m^2). For an ideal fluid all the particles move to the right with a velocity of 'C' (m/s). As the fluid flow

to the right with a velocity of C m/s, 'C' m of fluid pass section 1 every second. i.e. the volume of fluid passing section-1 every second is A x C.

$$\dot{v} = A \cdot C$$

Where:

- $\dot{v}$: Volume flow rate, m^3/s
- A: Cross-sectional area of the flow, m^2
- C: Mean (average) velocity of the fluid, m/s

Consider a real fluid flowing in the same pipe. Here particle velocity will be a variable across the flow stream.

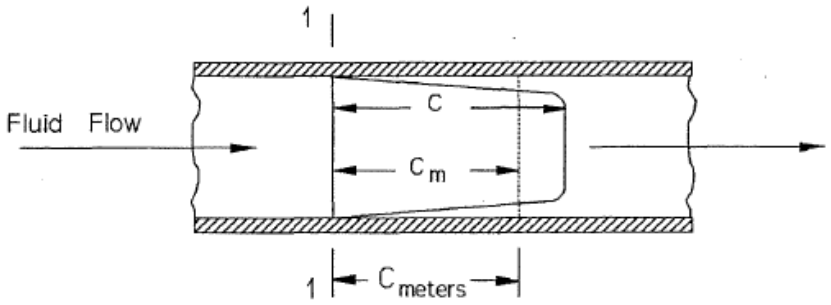


Figure 6.2: Mean velocity

If the mean velocity cm of all the fluid particles could be found, then similar to the ideal fluid.

$$Volume\ flow = area\ x\ mean\ velocity$$

Unless otherwise indicated, the velocity of flow is understood to refer to the average or mean velocity of all the particles in a flowing fluid.

6.2 Mass Flow Rate

Mass flow rate refers to the rate at which a fluid mass passes a given section in a flow stream. It is given in units of mass per unit of time, kilogram per second (kg/s). Mass flow rate is easily calculated from volume flow rate as follows.

$$\dot{m} = \rho \cdot \dot{v}$$

Where:

- $\dot{m}$: mass flow rate, kg/s
- ρ: density, kg/m^3
- $\dot{v}$: volume flow, m^3/s

Or

$$\dot{m} = \rho \cdot A \cdot C$$

Where:

- $\dot{m}$: mass flow rate, kg/s
- ρ: density, kg/m^3
- A: Cross-sectional area of the flow, m^2
- C: Mean (average) velocity of the fluid, m/s

Example 6.1. Oil of relative density 0.9 flows at full bore through a pipe with an internal diameter of 75 mm at a velocity of 1.2 m/s. Calculate the volume flow rate in cubic metres per second and the mass flow rate in tonnes per hour.

```
import math

# ------------------------------
# Given data
# ------------------------------
internal_diameter_mm = 75        # mm (pipe internal diameter)
velocity_m_per_s = 1.2           # m/s (fluid velocity)
relative_density = 0.9           # relative to water
density_water_kg_per_m3 = 1000   # standard density of water (kg/m³)

# ------------------------------
# Convert units and calculate radius
# ------------------------------
internal_diameter_m = internal_diameter_mm / 1000  # mm → m
radius_m = internal_diameter_m / 2                 # pipe radius (m)

# ------------------------------
# Cross-sectional area
# ------------------------------
area_m2 = math.pi * radius_m ** 2                # m²

# ------------------------------
# Volume and mass flow rates
# ------------------------------
volume_flow_rate_m3_per_s = area_m2 * velocity_m_per_s
```

```
# Density of oil (kg/m³)
density_oil_kg_per_m3 = relative_density * density_water_kg_per_m3

# Mass flow rate (kg/s)
mass_flow_rate_kg_per_s = density_oil_kg_per_m3 * volume_flow_rate_m3_per_s

# Mass flow rate in tonnes per hour
mass_flow_rate_tonnes_per_hour = (mass_flow_rate_kg_per_s * 3600) / 1000

# ----------------------------
# Output results
# ----------------------------
print(f"Volume flow rate, m³/s         : {volume_flow_rate_m3_per_s:.4f}")
print(f"Mass flow rate, tonnes/hour    : {mass_flow_rate_tonnes_per_hour:.4f}")
```

6.3 Flow Through Valves

Referring to the figure, the area of escape is the annular area of the circumferential opening between the valve and seat, which is the circumference multiplied by the lift.

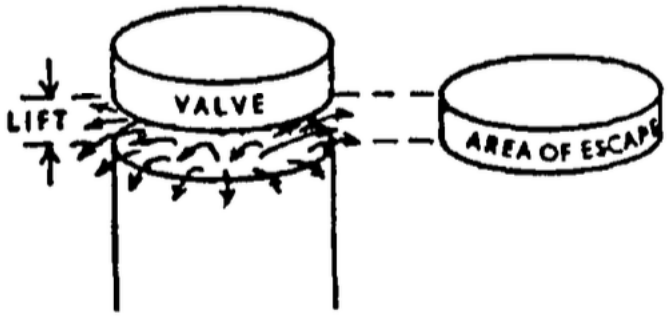


Figure 6.3: Flow through valves

The lift beyond which it would cause no restriction is when the circumferential area of the lift opening is equal to the cross-sectional area of the bore, thus,

$$\text{Circumference} \times \text{Lift} = \text{Cross-sectional area.}$$

$$\pi d \times L = \frac{\pi d^2}{4},$$

where L is the lift and d is the diameter of the valve.

Hence, a lift equal to one-quarter of the diameter of the valve allows full bore flow:

$$L = \frac{d}{4}.$$

Example 6.2. Calculate the volume flow rate of water in cubic metres per minute through a 150-millimetre diameter valve when the velocity of the water is 2.5 m/s and the valve lift is (i) 30 millimetres and (ii) 45 millimetres.

```
import math

# Given
d = 0.15                    # diameter in metres
V = 2.5                     # velocity in m/s

# Maximum effective lift
critical_lift_mm = (d / 4) * 1000

# Full bore area and flow rate
A_bore = math.pi * d**2 / 4
Q_full_m3_per_s = V * A_bore
Q_full = Q_full_m3_per_s * 60      # convert to m³/min


# (i) Lift = 30 mm
L1 = 0.030                  # lift in metres
A_annular1 = math.pi * d * L1
A_effective1 = min(A_annular1, A_bore)
Q1 = V * A_effective1 * 60


# (ii) Lift = 45 mm
L2 = 0.045                  # lift in metres
A_annular2 = math.pi * d * L2
A_effective2 = min(A_annular2, A_bore)
Q2 = V * A_effective2 * 60


print(f"Maximum effective lift: {critical_lift_mm:.4f} mm")
print(f"Full bore flow rate: {Q_full:.4f} m³/min")
print(f"At 30 mm lift: {Q1:.4f} m³/min")
print(f"At 45 mm lift: {Q2:.4f} m³/min")
```

6.4 Discharge Through an Orifice

When water discharges through an orifice in the side of a tank, the potential energy associated with the height of the water surface above the orifice is converted into kinetic energy of the efflux.

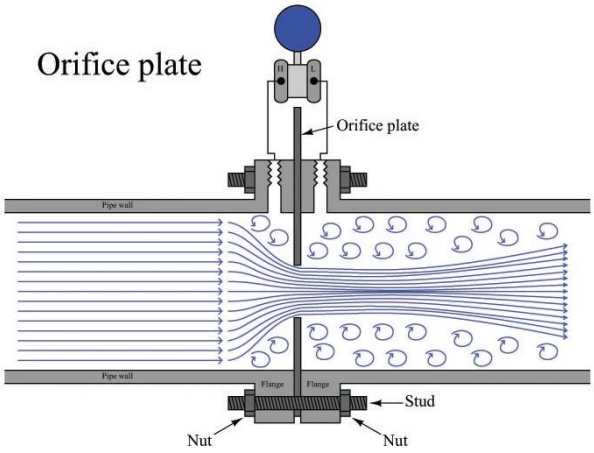


Figure 6.4: Orifice

This yields the theoretical velocity of the jet:

$$C = \sqrt{2gh}$$

where g is the acceleration due to gravity and h is the height of the water surface above the orifice.

The theoretical volume flow rate is then:

$$\dot{v}_{\text{theoretical}} = AC = A\sqrt{2gh}$$

where A is the area of the orifice.

Due to viscous effects, the actual velocity is less than the theoretical value. The coefficient of velocity, C_v, is defined as the ratio of actual velocity to theoretical velocity. Thus, the velocity-corrected flow rate is:

$$\dot{v}_{\text{velocity-corrected}} = C_v A\sqrt{2gh}$$

Downstream of a sharp-edged orifice, the jet contracts due to streamline curvature, forming a vena contracta. The vena contracta refers to the narrowest cross-section of a fluid jet shortly downstream of an orifice or nozzle, where the streamlines converge, resulting in the smallest area, maximum velocity, and minimum pressure.

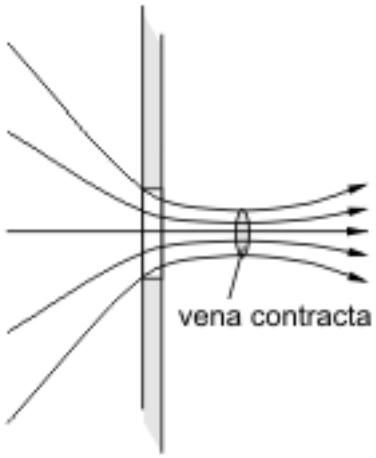


Figure 6.5: Vena contracta

The coefficient of contraction, C_A, is the ratio of the jet area at the vena contracta to the orifice area:

$$A_{\text{jet}} = C_A A$$

The actual volume flow rate therefore becomes:

$$\dot{v}_{\text{actual}} = C_A A \cdot C_v \sqrt{2gh}$$

The coefficient of discharge, C_d, is the ratio of the actual volume flow rate to the theoretical volume flow rate:

$$C_d = C_A \cdot C_v$$

Consequently,

$$\dot{v}_{\text{actual}} = C_d A \sqrt{2gh}$$

Example 6.3. Water escapes through a hole 20 mm in diameter in the side of a tank, with a water head of 3 m above the hole. Given a coefficient of velocity of 0.97 and a coefficient of reduction of area of 0.64, calculate (i) the velocity of the water jet as it exits the hole and (ii) the quantity of water escaping tonne per hour.

```
import math

# Given
```

```
diameter_mm = 20
diameter_m = diameter_mm / 1000
radius_m = diameter_m / 2
h = 3  # m
Cv = 0.97
Ca = 0.64
g = 9.81  # m/s²
density_water = 1000  # kg/m³

# Calculations
area = math.pi * radius_m ** 2

v_theoretical = math.sqrt(2 * g * h)
v_actual = Cv * v_theoretical

volume_flow_actual = Ca * area * v_actual  # m³/s

mass_flow = density_water * volume_flow_actual  # kg/s

mass_per_hour_kg = mass_flow * 3600
mass_per_hour_tonne = mass_per_hour_kg / 1000

# Output results
print(f"(i) Actual velocity of the water jet: {v_actual:.3f} m/s")

print(f"\n(ii) Quantity of water escaping per hour:")
print(f"    Volume flow rate: {volume_flow_actual:.6f} m³/s")
print(f"    Mass flow rate: {mass_flow:.4f} kg/s")
print(f"    Mass per hour: {mass_per_hour_kg:.4f} kg/hour")
print(f"    Mass per hour: {mass_per_hour_tonne:.4f} tonnes/hour")
```

6.5 Continuity Equation

Continuous flow exists in a flow system when the mass flow rate is constant throughout the system. If in the diagram below, the mass flow rate at 1 is equal to that at 2, then continuity exists.

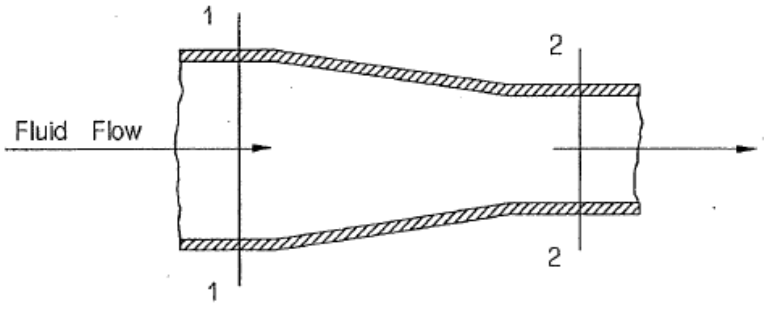


Figure 6.6: Continuity equation

Since $\dot{m}_1 = \dot{m}_2$, therefore

$$\rho_1 \cdot A_1 \cdot C_1 = \rho_2 \cdot A_2 \cdot C_2$$

If the fluid is incompressible (most liquids), then density will remain constant ($\rho_1 = \rho_2$) and the above equations may be written as:

$$A_1 \cdot C_1 = A_2 \cdot C_2$$

Or

$$\dot{v}_1 = \dot{v}_2$$

That is, the volume flow rate is constant for an incompressible fluid.

Example 6.4. A pipe decreases in diameter from 300 mm to 200 mm. Water flows from the larger to the smaller pipe at a constant rate of 18.4 kL/min. Calculate the mass flow rate and the velocities in the larger and smaller pipe.

```
import math

# Given
flow_rate_kL_per_min = 18.4   # Volumetric flow rate in kL/min
diameter_large_mm = 300       # Diameter of larger pipe in mm
diameter_small_mm = 200       # Diameter of smaller pipe in mm
density_water = 1000          # Density of water in kg/m³

# Convert volumetric flow rate to m³/s
flow_rate_m3_per_s = (flow_rate_kL_per_min / 60)

# Convert diameters to meters and calculate radii
d1 = diameter_large_mm / 1000  # m
d2 = diameter_small_mm / 1000  # m
```

```
r1 = d1 / 2
r2 = d2 / 2

# Cross-sectional areas
A1 = math.pi * r1**2  # m²
A2 = math.pi * r2**2  # m²

# Velocities
v1 = flow_rate_m3_per_s / A1  # m/s (larger pipe)
v2 = flow_rate_m3_per_s / A2  # m/s (smaller pipe)

# Mass flow rate
mass_flow_rate = density_water * flow_rate_m3_per_s  # kg/s

# Output results
print(f"Mass flow rate: {mass_flow_rate:.4f} kg/s")
print(f"Velocity in larger pipe (300 mm): {v1:.4f} m/s")
print(f"Velocity in smaller pipe (200 mm): {v2:.4f} m/s")
```

6.6 The Energy Equation for an Ideal Fluid

The steady flow equation is developed from the Law of Conservation of Energy. If there are no energy losses or energy additions in a flow system, then the total energy of a flowing fluid will remain constant.

A flowing fluid may lose energy as a result of fluid friction, heat energy transfer and fluid motors. For an ideal fluid, frictional losses are equal to zero. Energy may be added to a fluid via a pump or heat energy addition.

The total energy possessed by a flowing fluid consists of:

- internal energy
- potential energy
- kinetic energy and
- pressure energy (flow energy).

Generally in fluid mechanics the change in internal energy is considered to be negligible. Heat energy transfers are usually not considered in fluid mechanics and the frictional heat developed by a flowing fluid is relatively small. Therefore, the internal energy term is omitted from the energy balance.

Consider the flow system shown below. For this system, assume:

- an ideal fluid
- that there are no energy losses or additions and
- steady flow conditions exist.

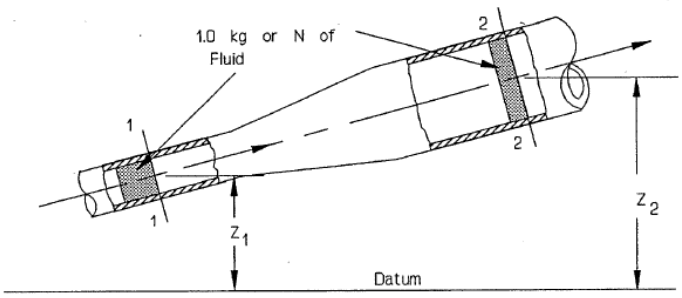


Figure 6.7: Energy equation

From the Law of Conservation of Energy:

The Total Energy at section 1 = The Total Energy at section 2

$$mZ_1g + \frac{mC_1^2}{2} + mP_1v_1 = mZ_2g + \frac{mC_2^2}{2} + mP_2v_2$$

Dividing through by mg:

$$\frac{mZ_1g}{mg} + \frac{mC_1^2}{2mg} + \frac{mP_1v_1}{mg} = \frac{mZ_2g}{mg} + \frac{mC_2^2}{2mg} + \frac{mP_2v_2}{mg}$$

Therefore

$$Z_1 + \frac{C_1^2}{2g} + \frac{P_1v_1}{g} = Z_2 + \frac{C_2^2}{2g} + \frac{P_2v_2}{g}$$

In fluid mechanics density (ρ) is generally used in preference to specific volume, i.e. $v = \frac{1}{\rho}$ Therefore:

$$Z_1 + \frac{C_1^2}{2g} + \frac{P_1}{g\rho_1} = Z_2 + \frac{C_2^2}{2g} + \frac{P_2}{g\rho_2}$$

In fluid mechanics, specific weight represents the force exerted by gravity on a unit volume of a fluid. For this reason, units are expressed as force per unit volume (e.g., N/m^3)

Specific weight is given $\gamma = g\rho$

Where:

γ: specific weight, $\mathrm{N/m^3}$

g: gravitational acceleration, $\mathrm{m/s^2}$

ρ: density, kg/m^3

$$Z_1 + \frac{C_1^2}{2g} + \frac{P_1}{\gamma_1} = Z_2 + \frac{C_2^2}{2g} + \frac{P_2}{\gamma_2}$$

Each term has units of m, therefore:

- Potential energy Z is known as the elevation head.
- Kinetic energy $\frac{c^2}{2g}$ is known as the velocity head.
- Pressure energy $\frac{P}{\gamma}$ is known as the pressure head.

Similarly, the total energy of a flowing fluid is known as the total head (H).

$$Total\ Head = Elevation\ Head + Velocity\ Head + Pressure\ Head$$

$$H = Z + \frac{C^2}{2g} + \frac{P}{\gamma}$$

The total head (H) will be a constant throughout a flow system if:

1. frictional losses (head loss) are equal to zero
2. work energy is not added by a pump (pump head) or removed by a motor.

Example 6.5. Consider a simple flow system consisting of a varying cross-section pipe. Water flows through this system at a rate of 2000 L/min. As the pipe increases in elevation from 30 m to 36 m it decreases in diameter from 10 cm to 3.0 cm. If the pressure is 6.5 MPa at the 30 m elevation, what is the pressure at 36 m?

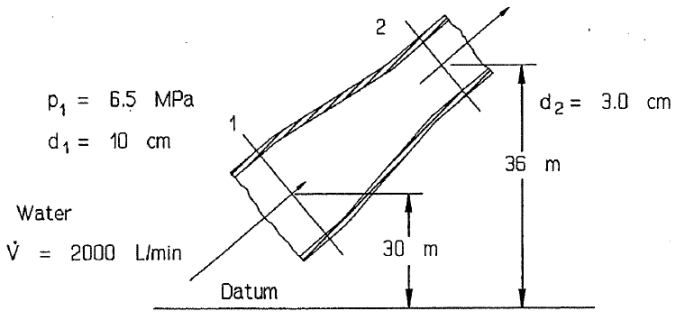


Figure 6.8: Example

```
import math

# Given
v_dot_lpm = 2000.0                # Flow rate, L/min
```

```
v_dot = v_dot_lpm / 60000.0       # Convert to m³/s
d1 = 0.10                         # Diameter at lower section, m
d2 = 0.03                         # Diameter at higher section, m
z1 = 30.0                         # Elevation at lower section, m
z2 = 36.0                         # Elevation at higher section, m
P1 = 6.5e6                        # Pressure at lower section, Pa
rho = 1000.0                      # Density of water, kg/m³

# Cross-sectional areas
A1 = math.pi * (d1 / 2.0)**2
A2 = math.pi * (d2 / 2.0)**2

# Velocities
c1 = v_dot / A1
c2 = v_dot / A2

# Bernoulli's equation to find P2
P2 = P1 + 0.5 * rho * (c1**2 - c2**2) + rho * g * (z1 - z2)

# Convert P2 to MPa
P2_MPa = P2 / 1e6

# Output results
print(f"Flow rate: {v_dot:.5f} m³/s")
print(f"Area 1: {A1:.6f} m²")
print(f"Area 2: {A2:.6f} m²")
print(f"Velocity 1: {c1:.4f} m/s")
print(f"Velocity 2: {c2:.4f} m/s")
print(f"Pressure at 36 m elevation: {P2_MPa:.4f} MPa")
```

6.7 Bernoulli's Equation

The Bernoulli's equation between two points in a fluid flow is given by:

$$P_1 + \frac{1}{2}\rho C_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho C_2^2 + \rho g h_2$$

Where:

- P_1 and P_2 are the pressures at points 1 and 2, respectively.
- ρ is the density of the fluid.
- C_1 and C_2 are the velocities of the fluid at points 1 and 2, respectively.
- g is the acceleration due to gravity.

- h_1 and h_2 are the heights of the fluid at points 1 and 2, respectively.

6.8 Venturi Meter

A venturi meter measures liquid flow rates in pipelines. The device features a pipe section that narrows in the middle (called the throat) and widens at both ends, as shown below:

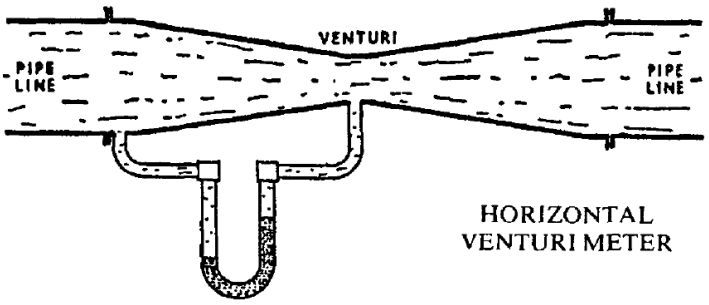


Figure 6.9: Venturi meter

When the entrance and throat areas are known, and the pressure readings (or pressure difference) at these points are measured, Bernoulli's equation can be used to calculate the liquid velocity and flow rate. Typically, a venturi meter is installed horizontally in the pipeline, which simplifies calculations since there is no elevation difference between points ($Z_1 = Z_2$), thus eliminating the height terms.

Example 6.6. Calculate the mass flow rate in kilograms per second through a smooth, horizontal venturi meter with an entrance diameter of 375 mm and a throat diameter of 125 mm, given a pressure difference of 457 mm of water

```
import math

# Given
D1 = 0.375  # Entrance diameter in meters
D2 = 0.125  # Throat diameter in meters
h = 0.457   # Pressure head difference in meters of water
g = 9.81    # Gravitational acceleration in m/s²
rho = 1000  # Density of water in kg/m³

# Calculate cross-sectional areas
A1 = math.pi * (D1 / 2)**2
A2 = math.pi * (D2 / 2)**2

# Compute the squared area ratio (A2/A1)^2
```

```
ratio_squared = (A2 / A1)**2

# Calculate throat velocity c2
c2 = math.sqrt((2 * g * h) / (1 - ratio_squared))

# Calculate volumetric flow rate v_dot using throat section
v_dot = A2 * c2

# Calculate mass flow rate
mass_flow_rate = rho * v_dot

# Output results
print(f"Entrance area: {A1:.6f} m²")
print(f"Throat area: {A2:.6f} m²")
print(f"Throat velocity: {c2:.4f} m/s")
print(f"Volumetric flow rate: {v_dot:.4f} m³/s")
print(f"Mass flow rate: {mass_flow_rate:.4f} kg/s")
```

Example 6.7. A horizontal venturi meter has an inlet diameter of 450 mm and a throat diameter of 225 mm. The pressure difference between these two points is equivalent to 381 mm of water. Given the density of fresh water is 1000 kg/m³, calculate the mass flow rate through the meter.

```
import math

# Given
D1 = 0.450  # Inlet diameter in meters
D2 = 0.225  # Throat diameter in meters
h = 0.381   # Pressure head difference in meters of water
g = 9.81    # Gravitational acceleration in m/s²
rho = 1000  # Density of water in kg/m³

# Calculate cross-sectional areas
A1 = math.pi * (D1 / 2)**2
A2 = math.pi * (D2 / 2)**2

# Compute the squared area ratio (A2/A1)^2
ratio_squared = (A2 / A1)**2

# Calculate throat velocity c2
c2 = math.sqrt((2 * g * h) / (1 - ratio_squared))

# Calculate volumetric flow rate v_dot using throat section
v_dot = A2 * c2

# Calculate mass flow rate
```

```
mass_flow_rate = rho * v_dot

# Output results
print(f"Entrance area: {A1:.6f} m²")
print(f"Throat area: {A2:.6f} m²")
print(f"Throat velocity: {c2:.4f} m/s")
print(f"Volumetric flow rate: {v_dot:.4f} m³/s")
print(f"Mass flow rate: {mass_flow_rate:.4f} kg/s")
```

Example 6.8. A 300 mm diameter pipe has a venturi meter with a throat diameter of 100 mm. A U-tube gauge filled with water and mercury measures a pressure head difference of 250 mm between the inlet and throat. The meter coefficient is 0.95. Using the density of water (1000 kg/m³), calculate the discharge rate through the pipe in m³/s.

```
import math

# ------------------------------
# Given data
# ------------------------------
D1 = 0.3  # Inlet diameter in meters
D2 = 0.1  # Throat diameter in meters
Cd = 0.95  # Coefficient of discharge
h_mercury = 0.250  # Manometer reading in meters of mercury
SG = 13.6  # Specific gravity of mercury relative to water
g = 9.81  # Acceleration due to gravity in m/s²

# ------------------------------
# Cross-sectional areas
# ------------------------------
A1 = math.pi * (D1**2) / 4  # Area at the inlet
A2 = math.pi * (D2**2) / 4  # Area at the throat

# ------------------------------
# Equivalent water head
# ------------------------------
# The difference in water levels (equivalent head) generates
# a hydrostatic pressure in the connecting legs that counteracts
# the mercury column. Multiplying by (SG - 1) accounts for the
# specific gravity of mercury relative to water.
h_eq = h_mercury * (SG - 1)

# ------------------------------
# Diameter ratio
# ------------------------------
beta = D2 / D1
```

```
# ----------------------------
# Discharge calculations
# ----------------------------
# Theoretical discharge based on the ideal flow equation
Q_theoretical = A2 * math.sqrt((2 * g * h_eq) / (1 - beta**4))

# Actual discharge accounting for the coefficient of discharge
Q_actual = Cd * Q_theoretical

# ----------------------------
# Output results
# ----------------------------
print(f"Cross-sectional area at inlet (A1): {A1:.4f} m²")
print(f"Cross-sectional area at throat (A2): {A2:.4f} m²")
print(f"Equivalent water head (h_eq): {h_eq:.4f} m")
print(f"Theoretical discharge: {Q_theoretical:.4f} m³/s")
print(f"Actual discharge: {Q_actual:.4f} m³/s")
```

6.9 Power Required for Pumping Water

The power needed to pump water is the rate at which work is done to increase the fluid's potential energy by lifting it against gravity, often with friction or other losses.

In ideal conditions, the theoretical power P is calculated as

$$P = \rho g h \dot{v}$$

where ρ is the fluid density, g is gravitational acceleration, h is the total head, and $\dot{v}$ is the volumetric flow rate. This is derived from the mass flow rate $\dot{m} = \rho \dot{v}$.

Actual input power accounts for pump efficiency η, given by

$$\eta = \frac{P_{\text{output}}}{P_{\text{input}}}$$

Example 6.9. Water is pumped to a height of 20 m at a rate of 12 litres per second. If the pump efficiency is 75%, calculate the input power.

```
# Given
flow_rate_lps = 12.0    # litres per second
height = 20.0           # metres
efficiency = 0.75
rho = 1000.0            # density of water in kg/m³
```

```
g = 9.81                # acceleration due to gravity in m/s²

# Calculations
v_dot = flow_rate_lps / 1000.0                  # volumetric flow rate in m³/s
m_dot = rho * v_dot                             #  mass flow rate in kg/s
output_power = m_dot * g * height           # watts
input_power = output_power / efficiency     # watts

# Result
print(f"Actual input power (with 75% efficiency): {input_power:.4f} W ({input_power / 1
```

Example 6.10. A water pump operates with an actual input power of 3 kW and an overall efficiency of 70%. The pump delivers water at a rate of 12 litres per second. Assuming the power is used solely to increase the gravitational potential energy (negligible velocity and pressure heads), calculate the height (head) to which the water is pumped.

```
# Given
input_power = 3000.0    # Actual input power in W
efficiency = 0.70       # Pump efficiency
flow_rate_lps = 12.0    # Flow rate in litres per second
rho = 1000.0            # Density of water in kg/m³
g = 9.81                # Gravitational acceleration in m/s²

# Volumetric flow rate v_dot (m³/s)
v_dot = flow_rate_lps / 1000.0

# Mass flow rate m_dot (kg/s)
m_dot = rho * v_dot

# Output power delivered to the water (W)
output_power = input_power * efficiency

# Head h (m) using P = m_dot * g * h
h = output_power / (m_dot * g)

# Result
print(f"Head (height pumped): {h:.2f} m")
```

References

Bolton, W. (2021). *Engineering science* (Seventh edition). Routledge.

Hannah, J., & Hillier, M. J. (1995). *Applied mechanics* (3rd edition). Longman Pub Group.

Russell, P. A., Jackson, L., & Embleton, W. (2021). *Applied mechanics for marine engineers* (7th edition). Reeds.

Shaw, Z. (2017). *Learn python 3 the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code* (4th edition). Addison-Wesley Professional.

Sweigart, A. (2020). *Automate the boring stuff with python, 2nd edition: Practical programming for total beginners*. No Starch Press.

Sweigart, A. (2021). *Beyond the basic stuff with python: Best practices for writing clean code*. No Starch Press.

Colophon

This booklet was typeset with Quarto v.1.8.26.

Appendix A

Greek Letters

The following tables present the names of Greek letters and selected symbols commonly used in engineering courses, ensuring precise reference and avoiding reliance on informal descriptors such as "squiggle."

Table A.1: Greek letters.

Lower Case	Upper Case	Name
α	A	alpha
β	B	beta
γ	Γ	gamma
δ	Δ	delta
ϵ	E	epsilon
ζ	Z	zeta
η	E	eta
θ	Θ	theta
ι	I	iota
κ	K	kappa
λ	Λ	lambda
μ	M	mu
ν	N	nu
ξ	Ξ	xi
o	O	omicron
π	Π	pi
ρ	P	rho
σ	Σ	sigma
τ	T	tau
υ	Υ	upsilon
ϕ	Φ	phi
χ	X	chi

Lower Case	Upper Case	Name
ψ	Ψ	psi
ω	Ω	omega

Table A.2: Commonly used symbols in engineering courses.

Symbol	Name	Use	Course
Δ	Delta	Change	Thermodynamics
Δ	Delta	Displacement	Naval Architecture
∇	Nabla	Volume	Naval Architecture
Σ	Sigma	Sum	Thermodynamics, Naval Architecture, Applied Mechanics
σ	Sigma	Stress	Thermodynamics, Applied Mechanics
ϵ	Epsilon	Modulus of elasticity	Thermodynamics, Applied Mechanics
η	Eta	Efficiency	Thermodynamics
μ	Mu	Friction	Thermodynamics, Applied Mechanics
ω	Omega	Angular velocity	Thermodynamics, Applied Mechanics
ρ	Rho	Density	Thermodynamics, Naval Architecture
τ	Tau	Torque	Thermodynamics, Applied Mechanics

Appendix B

Revision History

Table B.1: Changelog.

Version	Date	Description
2.0	2026-01-	Major revision with new chapters
1.0	2025-09-	Updated SI units section
0.2	2025-09-03	Added formulae section
0.1	2025-09-01	Initial development

Index