Modelling Terminal Velocity for Different Size Spheres in a Wind Tunnel from Dimensional Analysis

Course code:	ENME314
Lab title:	LabB: Dimensional Analysis
Author:	Justin France
Student number:	75349339
Student email:	Jfr125@uclive.ac.nz
Date:	07/05/2025

Introduction

Experiments were carried out to determine and model the terminal velocity of 20mm lead spheres via the approach of dimensional analysis, due to the wind speed limitations of equipment and the impracticality of carrying out a real-world testing. The approach of dimensionless analysis was carried out with 50mm and 202mm spheres over a range of wind speeds which was around the value for theoretically calculated terminal velocities.

The approach of geometric similarities, dynamic and kinematic similarities and wind tunnel drag corrections was used to model this system which will be compared against the Clift standard sphere data. Comparisons between the raw calculated drag coefficients and the application of wind tunnel theory methods was undertaken to validate the improvement on the accuracy of the measured data. Verification of any assumptions required during analysis will be checked for robustness to validate the terminal velocity of a lead sphere.

Background

Derivation for terminal velocity:

From newtons second law $\sum F = ma = 0$ occurs when the drag force balances out the weight force. Under this condition the velocity for a falling object is at maximum speed known as terminal velocity.

$$mg = F_d \qquad mg = \frac{1}{2}A\rho V_t^2 C_d$$

Rearranging for V_t then substituting in for the area of a sphere.

$$V = \sqrt{\frac{2mg}{A\rho C_d}} \qquad V = \sqrt{\frac{2mg}{\frac{\pi D^2}{4}\rho C_d}}$$

Theoretical Preliminary workings:

For a sphere with $D = 0.02\ [m]$ and with $c_d = 0.48$

$$V_t = \sqrt{\frac{2m_{pb}g}{\rho_{air}\left(\frac{1}{4}\pi D^2\right)C_d}} = \sqrt{\frac{2*0.0565*9.81}{1.202\left(\frac{1}{4}\pi 0.02^2\right)*0.48}} = 78.20\ [\tfrac{m}{s}]$$

$$R_e = \frac{\rho V_t D}{\mu} = \frac{1.202*78.2*0.02}{1.81*10^{-5}} = 103863$$

knowns	
T	= 293.15[k]
P_{ATM}	=101325 $[Pa]$
R_d'	=287.6 $[JK^{-1}kg^{-1}]$
ρ_{air}	= 1.202 $[\frac{kg}{m^3}]$
m_{pb}	= 0.0565 $[kg]$
μ	= $1.81 * 10^{-5}$ [Ns/m^2]

Reverse engineer for theoretical terminal velocities for 50mm and 202mm.

50[mm] sphere with $Re = 103863$ $\qquad V_t = \frac{R_e*\mu}{\rho_{air}*D} = \frac{101738\ *1.81*10^{-5}}{1.202*0.05} = 31.28\ \left[\frac{m}{s}\right]$

202[mm] sphere with $Re = 101738$ $\qquad V_t = \frac{R_e*\mu}{\rho_{air}*D} = \frac{101738\ *1.81*10^{-5}}{1.202*0.202} = 7.74\ \left[\frac{m}{s}\right]$

Buckingham-PI method:

Name	Symbol	Unit [SI]	Dimension
Drag Force	F_d	Newtons	MLT^{-2}
mass	m	Kg	M
length	D	M	L
Time	T	s	T
Density	ρ	Kgm^{-3}	ML^{-3}
Viscosity	μ	$Pa.s\ =\ kgm^{-1}s^{-1}m$	$ML^{-1}T^{-1}$
gravity	g	ms^{-2}	LT^{-2}
Terminal velocity	V_t	ms^{-1}	LT^{-1}

Buckingham PI for terminal velocity:

$$V_t = \sqrt{\frac{2mg}{\rho\left(\frac{1}{4}\pi D^2\right)C_d}} = \frac{MLT^{-2}}{ML^{-3}L^{-2}}$$

Number of variables $\boldsymbol{c_d = f(F_d, D, V_t, \rho, \mu)}$ $\therefore n = 5$

Number of dimensions in the problem$(\boldsymbol{M, T, L})$ $\therefore j = 3$

Number of dimensionless quantities $k = n - j$

$\therefore k = 5 - 3 = 2$

$$\Pi_i = F^a + D^b + V_t^c + \rho^d$$

$$\Pi_i = (MLT^{-2})^a + (L)^b + (LT^{-1})^c + (ML^{-3})^d$$

Let $a = 1$ for the dependent variable F_d

$M:\ 0 = 1 + d$ $\qquad \therefore d = -1$

$T: 0 = -2 * 1 - c$ $\qquad \therefore c = -2$

$L: 0 = 1 + b + c - 3d$ $\qquad \therefore b = -2$

$$\Pi_i = F^1 + D^{-2} + V_t^{-2} + \rho^{-1}$$

$$\Pi_i = \frac{F^1}{D^2V_t^2\rho^1}$$

$$\Pi_i = \frac{F^1}{\frac{\pi}{4}D^2V_t^2\rho^1}$$

From the Bucking pi method, it was determined that, the drag coefficient is unitless dimension.

$$\Pi_i = \frac{F^1}{\frac{\pi}{4}D^2V_t^2\rho^1} \qquad \frac{MLT^{-2}}{L^2ML^{-3}L^2T^{-2}} \qquad C_d = \frac{F_d}{\frac{\pi D^2}{4}\rho V_t^2}$$

Buckingham PI for Reynalds number:

$$V_t = \frac{Re\mu}{D\rho} = \frac{ML^{-1}T^{-1}}{LML^{-3}}$$

Number of variables $\boldsymbol{Re = f(\rho, V_t, D, \mu)}$ $\quad \therefore \boldsymbol{n = 5}$

Number of dimensions in the problem $\boldsymbol{(M, T, L)}$ $\quad \therefore \boldsymbol{j = 3}$

Number of dimensionless quantities $\quad \boldsymbol{k = n - j}$

$\therefore \boldsymbol{k = 5 - 3 = 2}$

$$\Pi_i = (\rho)^a + (v_t)^b + (D)^c + (\mu)^d$$

$$\Pi_i = (ML^{-3})^a + (LT^{-1})^b + (L)^c + (ML^{-1}T^{-1})^d$$

Let $a = 1$ for the independent variable ρ

M: $0 = 1 + d \quad \therefore d = -1$

T: $0 = -b - d \quad \therefore b = 1$

L: $0 = -3 * 1 + b + c - d \quad \therefore c = 1$

$$\Pi_i = (\rho)^1 + (v_t)^1 + (D)^1 + (\mu)^{-1}$$

$$\Pi_i = \frac{\rho^1 V_t^1 D^1}{\mu^1}$$

From the Buckingham-pi it is determined that Reynolds number is a unitless dimension.

$$\Pi_i = \frac{\rho^1 V_t^1 D^1}{\mu^1} \qquad \frac{ML^{-3}LT^{-1}L}{ML^{-1}T^{-1}} \qquad R_e = \frac{\rho V_t D}{\mu}$$

Therefore, the fluid drag force has a relationship with the dimensionless parameters of Reynolds number and drag coefficient in the form of the function $F_d = f(C_d, Re)$.

Equipment description:

An aeronautical closed loop wind tunnel with a six-axis force balance with an attached string was used for simulation of a 20mm free falling lead shot, as described by the ENME314 lab manual. Due to the equipment limitations for only producing linear air flow at speeds between 7.0 m/s and 50 m/s scaled models were manufactured to enable, testing within the wind tunnel, operating parameters. If was assumed the sensors for the wind speed and force balance are calibrated and reading correctly during the experiment. It was assumed the dimensional measurements provided for the wind tunnel walls are correct.

Results

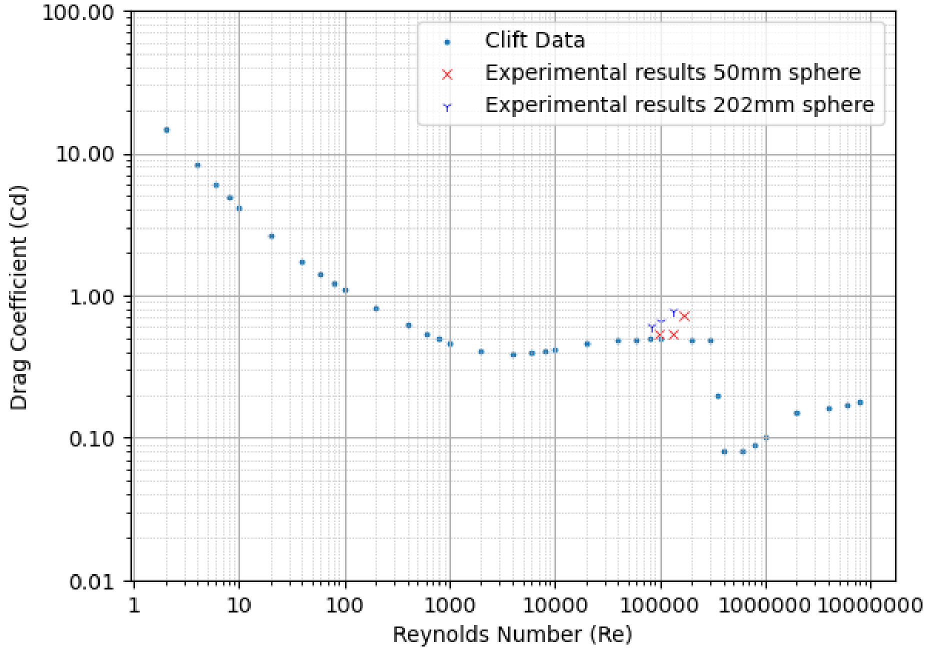

Figure 2: Log plot showing the relationship between the experiment and standard Clift data

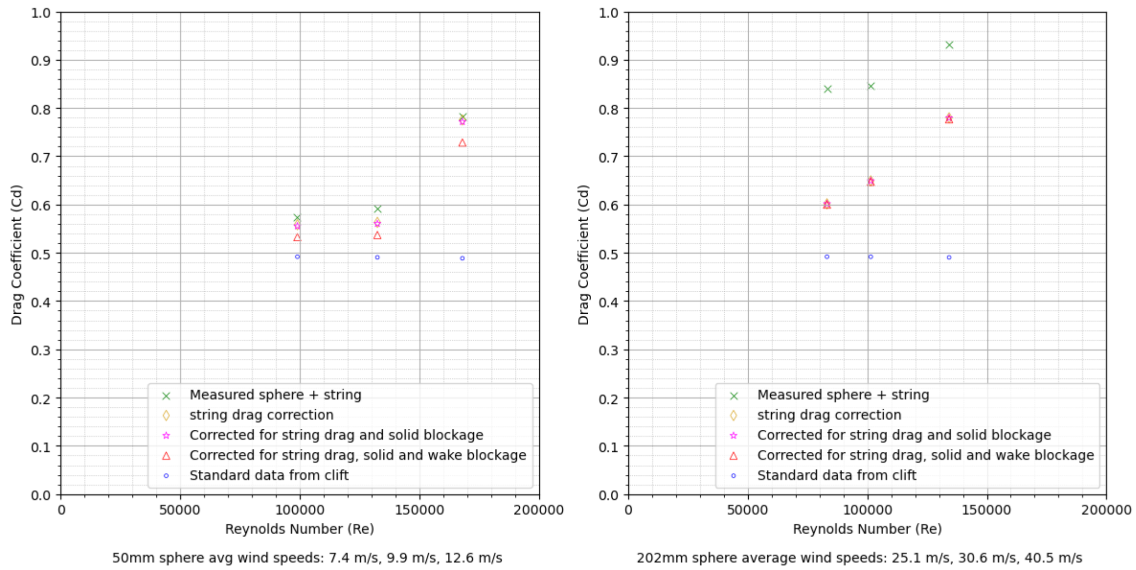

Figure 1: Linear plots for the 50mm and 202mm sphere at different wind speeds and standard Clift data

Table1: Mach number from the averaged data compressibility effects

Mach number	Wind velocity
0.0215	7.4 m/s
0.0288	9.9 m/s
0.0365	12.6 m/s
0.0731	25.1 m/s
0.0892	30.6 m/s
0.118	40.5m/s

Tabel2: Comparing sphere+string agents string drags averaged data drag coefficients

Sphere+string c_d	String c_d
0.5734	0.8026
0.5914	1.0144
0.7832	0.903
0.8392	1.0562
0.845	1.024
0.9315	1.0703

Discussion

Assumptions:

To validate dynamic similarity assumption the values for Mach number of 0.3 or less it was assumed that the air was incompressible. This assumption for compressibility was not exactly valid as there was a small amount of compressibility which increased proportionally with wind velocity as shown by table 1.

The kinematic similarity condition is only partially satisfied due to requirement of the wind tunnel experiment requiring a string which makes this assumption for a smooth sphere incorrect. To rectify this the string drag was measured and then subtracted off the sphere and string drag. This helped lower the sphere drag values but was deemed as mediocre method as the exact conditions like wind speeds could not be replicated between test. As seen by table 2 the averaged drag coefficients values obtained from string are typically 0.3 higher than the string+sphere and therefore the string experiments are placing a bias on the final drag coefficient results.

The wind tunnel will not exactly simulate the conditions of a lead shot tower as the proximity of the wind tunnel walls is closer than a shot tower. This configuration of the air tunnel walls being closer in proximity to the spheres changes the wind flow pattern around the spheres and causes the air to become accelerated. This was known as solid blockage and was a condition which changes the pressure on the frontal face of the sphere and therefore the air drag. To account for this string blockage Calculation was applied which lowered the c_d values. The wake blockage behind the spheres also effects the flow pattern around the sphere and was correct for which further improved the drag coefficients.

It was assumed a perfectly symmetric sphere and the air density remains the same throughout the duration of all the experiments.

Experiment comments

The recorded data seen from figures 1 and 2 show that the drag coefficients do not satisfy the known values from the standard Clift data for smooth spheres. With this experimental approach discrepancy between the string + sphere and string drag coefficient values places to much of a bias on the sting data set. It seems that spheres with larger frontal surface areas and higher air speeds on the sphere has a greater impact of the accuracy on the results of cd. Therefore, it is valid to say that sting drag, solid blockage correction and wake correction does improve the accuracy of the results. Even though the assumptions for dynamics and kinematic similarities may not be entirely valid it is still an improvement over the raw data.

Effects of sphere surface roughness

When rough spheres are used the drag forces become a function of the dimensionless parameters, $F_d = f(C_d, Re, Rr)$. Where the additional parameter of relative roughness $Rr = \frac{\varepsilon}{D}$ has a strong dependence at higher Reynolds numbers which causes the step in the plot to occur sooner which is known as the drag crisis. The shape of the curve and location of the step is dependent on the surface roughness of the sphere.

Conclusion

To obtain comparable results to the Clift standard data set the geometric, dynamic and kinematic similarities assumptions must be valid, therefore these experiments would not accurately model a 20mm lead sphere free falling in a shot tower. When comparing the experimental collected data to Clift standard data set it is valid to say that applying string correction, solid blockage correction, wake blockage correction improves the accuracy of the data but not to the point where it was considered usable. The assumptions that the dynamic and kinematic was valid is not true due to compressibility effects, the differences between the experimental conditions such as wind speed between sphere+string and string that resulted in a bias sting c_d data set. Flow separation effects were not fully accounted for which by the solid, wake blockage and sting correlation equations and as air flow speeds increased the accuracy of the drag coefficients also increased. Therefore, the most accurate data point tended to be the at speeds of under 10m/s with the 50mm sphere.

Appendix:

University of Canterbury, ENME314, lecture notes.

University of Canterbury, ENME314, Laboratory Manual 2023.

Code for data processing and plots:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter



def load_clift_data(std_data):
    clift_data = np.loadtxt(std_data, delimiter=",", skiprows=1)
    return clift_data

def load_files(data_set):
    """ load in data from a cvs file cap file data to 100 data points"""
    data = np.loadtxt(data_set, delimiter="\t")

    data = data[:100]
    time = np.array(data[:,[0]])
    probe_velocity = np.array(data[:,[1]])
    side_force =np.array(data[:,[2]])
    drag_force = np.array(data[:,[3]])
    lift_force = np.array(data[:,[4]])
    x_moment = np.array(data[:,[5]])
    y_moment = np.array(data[:,[6]])
    z_moment = np.array(data[:,[7]])
    #print("time\n\n",time)
    return time, probe_velocity, side_force, drag_force, lift_force, x_moment, y_moment, z_moment


def caculate_Cd(data_sphere, data_string, density, Diameter, Area_tot, Area_string):
    """caculate the drag cofficient"""
    sphere_data = load_files(data_sphere)
    string_data = load_files(data_string)

    velocity_sphere = sphere_data[1]
    drag_force_sphere = -1*sphere_data[3]

    velocity_string = string_data[1]
    drag_force_string = -1*string_data[3]

    Cd_sphere = drag_force_sphere / (0.5 * density * velocity_sphere**2 * Area_tot)
    Cd_string = drag_force_string / (0.5 * density * velocity_sphere**2 * Area_string)

    Cd_sphere_average =  np.sum(Cd_sphere)/ np.size(Cd_sphere)
    Cd_string_average =  np.sum(Cd_string)/ np.size(Cd_string)



    # print(drag_force_string.shape)
    #print(velocity_sphere.shape)

    #print(drag_force_sphere)
    #print(velocity_sphere)
    #print(density)
    #print(Area_tot)
    #print(Cd_sphere)
    #print(drag_force_string)
    #print(velocity_string)
```

```python
    #print(drag_force_string)
    #print(velocity_string)
    #print(Area_string)
    #print(Cd_string)

    #print(Cd_string_average)
    #print(Cd_sphere_average)

    print(np.round(Cd_sphere_average,4), np.round(Cd_string_average,4) )
    return Cd_sphere, Cd_string, velocity_sphere


def caculate_Reynalds_number(data_sphere, data_string, Diameter_sphere, Diameter_string ,kinematic_viscosity):
    """caculate the reynalds number"""
    sphere_data = load_files(data_sphere)
    string_data = load_files(data_string)
    velocity_sphere = sphere_data[1]
    velocity_string = string_data[1]

    Re_sphere = (velocity_sphere * Diameter_sphere) / (kinematic_viscosity)
    Re_string = (velocity_string * Diameter_string) / (kinematic_viscosity)

    Re_sphere_average =  np.sum(Re_sphere)/ np.size(Re_sphere)
    Re_string_average =  np.sum(Re_string)/ np.size(Re_string)
    #print(velocity_sphere)
    #print(Diameter_sphere)
    #print(kinematic_viscosity)
    #print(Re_sphere)
    #print(velocity_sphere)
    #print(Diameter_string)
    #print(Re_string)

    #print(Re_string_average)
    #print(Re_sphere_average)
    return Re_sphere, Re_string

def string_drag_corrolation (data_sphere, data_string, density, sphere_frontal_area):
    sphere_data = load_files(data_sphere)
    string_data = load_files(data_string)
    drag_force_sphere = -1*sphere_data[3]
    drag_force_string = -1*string_data[3]
    velocity_sphere = sphere_data[1]

    drag_force = drag_force_sphere - drag_force_string
    CDs_rAs_r = (2 * drag_force_sphere)/(density * velocity_sphere ** 2)
    CD_r_A = ( 2 * drag_force_string) / (density * velocity_sphere **2 )
    corrected_Cd = (CDs_rAs_r - CD_r_A) / sphere_frontal_area
    #print(drag_force)
    #print(string_drag_correction)
    #print(CD_r_A)
    #print(corrected_Cd)
    return drag_force, CDs_rAs_r, CD_r_A, corrected_Cd
```

```python


def blockage_corrolation (data_sphere, data_string, speed_of_sound, Diameter_sphere, sphere_frontal_area, n_power,  k3s, H_tunnel, B_breadth, cross_sectional_area_tunnel, density):
    sphere_data = load_files(data_sphere)
    string_data = load_files(data_string)
    velocity_sphere = sphere_data[1]
    string_drag  = string_drag_corrolation (data_sphere, data_string, density, sphere_frontal_area)
    corrected_Cd = string_drag[3]

    Mach_number = velocity_sphere / speed_of_sound

    beta = (1- Mach_number ** 2) ** (1/2)

    epleson_3_s = (
        k3s / (beta ** 3)
        * (B_breadth / H_tunnel + H_tunnel / B_breadth) ** n_power
        * ((1/6 * np.pi * Diameter_sphere ** 3) / ( 2 * Diameter_sphere))** (1/2)
        * (sphere_frontal_area) / (cross_sectional_area_tunnel ** (3/2)) )

    Cd_sphere_corr_solid_blockage  =  corrected_Cd / ((1 + epleson_3_s) ** 2)

    theta_factor = 0.96+1.94 *np.exp(-0.06 * Diameter_sphere / Diameter_sphere)

    # Cd_sphere_wake_blockage = (np.sqrt(-1 + (1 + 4 * theta_factor * (sphere_frontal_area / cross_sectional_area_tunnel) * Cd_sphere_corr_solid_blockage))
    #                              / (2 * theta_factor * sphere_frontal_area / cross_sectional_area_tunnel))

    c_d_numerator = -1 + np.sqrt(1 + 4 * theta_factor * (sphere_frontal_area / cross_sectional_area_tunnel) * Cd_sphere_corr_solid_blockage)
    c_d_denominator = 2 * theta_factor * (sphere_frontal_area / cross_sectional_area_tunnel)
    Cd_sphere_wake_blockage = c_d_numerator / c_d_denominator

    Cd_sphere_wake_blockage_average =  np.sum(Cd_sphere_wake_blockage )/ (np.size(Cd_sphere_wake_blockage))

    #print(Cd_sphere_wake_blockage)

    #print(Mach_number)
    #print (np.round(np.mean(Mach_number),4))
    #print(beta)
    #print(epleson_3_s)
    #print(Cd_sphere_corr)
    #print(corrected_Cd)
    #print(theta_factor)
    #print(Cd_sphere_wake_blockage)

    # print(velocity_sphere)
    # print(Diameter_sphere)
    # print(k3s)
    # print(beta)
    #print((B_breadth))
    #print((H_tunnel))
    #print(n_power)
    #print(Diameter_sphere)
    #print(sphere_frontal_area)
    #print(cross_sectional_area_tunnel)
    return Mach_number, beta, epleson_3_s, Cd_sphere_corr_solid_blockage, theta_factor, Cd_sphere_wake_blockage,  Cd_sphere_wake_blockage_average
```

```python


def log_plot_Re_Cd(clift_data, density, exp_Re_values=None, exp_Cd_values=None):
    """Plot the Reynolds number and Cd values in log-log scale."""
    Re_clift = np.array(clift_data[:, 0])
    cd_clift = np.array(clift_data[:, 1])



    fig, ax = plt.subplots()
    ax.scatter(Re_clift, cd_clift, label="Clift Data", s=3)

    # Add experimental data if provided
    if exp_Re_values is not None and exp_Cd_values is not None:
        Re_50mm_sphere = exp_Re_values[:3]
        Cd_50mm_sphere = exp_Cd_values[:3]
        Re_202mm_sphere = exp_Re_values[3:]
        Cd_202mm_sphere = exp_Cd_values[3:]
        # print( Re_50mm_sphere)
        # print( Cd_50mm_sphere,"\n")
        # print(Re_202mm_sphere)
        # print(Cd_202mm_sphere,"\n")


        ax.scatter(Re_50mm_sphere, Cd_50mm_sphere, color='red', marker='x', label="Experimental results 50mm sphere", s=20, linewidth=0.5)
        ax.scatter(Re_202mm_sphere, Cd_202mm_sphere, color='blue', marker='1', label="Experimental results 202mm sphere", s=25, linewidth=0.5)


    ax.set_xscale("log")
    ax.set_yscale("log")
    ax.grid()
    ax.grid(which='minor', linestyle=':', linewidth=0.5)
    ax.set_xlabel("Reynolds Number (Re)")
    ax.set_ylabel("Drag Coefficient (Cd)")

    ax.get_xaxis().set_major_formatter(ScalarFormatter())
    ax.ticklabel_format(style='plain', axis='x')

    ax.set_yticks([0.01, 0.1, 1, 10, 100])
    ax.get_yaxis().set_major_formatter(ScalarFormatter())
    ax.ticklabel_format(style='plain', axis='y')
    ax.legend()
    plt.show()

```

```python
def linear_plot_Re_Cd(exp_Re,
                      experimental_Cd_string_sphere,
                      experimental_Cd_corrected_string_sphere,
                      experimental_Cd_solid_blockage,
                      experimental_Cd_wake_string_sphere,
                      Cs_smooth_sphere,
                      ave_velocity_list):
    """ plot the reynalds number aganist drag cofficient"""


    x_ticks = np.linspace(0, 200000, 5)
    y_ticks = np.linspace(0, 1, 11)


    velocities_50mm = ", ".join([f"{v} m/s" for v in ave_velocity_list[:3]])
    velocities_202mm = ", ".join([f"{v} m/s" for v in ave_velocity_list[3:]])

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
    # plots for 50mm sphere
    ax1.scatter(exp_Re[:3],  experimental_Cd_string_sphere[:3], color="green", marker="x", label="Measured sphere + string",  s=30, linewidths=0.5)
    ax1.scatter(exp_Re[:3], experimental_Cd_corrected_string_sphere[:3], marker="d", facecolors='none', edgecolors="goldenrod", label="string drag correction", s=30, linewidths=0.5)
    ax1.scatter(exp_Re[:3], experimental_Cd_solid_blockage[:3], marker="*",facecolors='none', edgecolors="magenta", label="Corrected for string drag and solid blockage", s=30, linewidths=0.5)
    ax1.scatter(exp_Re[:3], experimental_Cd_wake_string_sphere[:3], marker="^",facecolors='none', edgecolors="red", label="Corrected for string drag, solid and wake blockage", s=30, linewidths=0.5)
    ax1.scatter(exp_Re[:3], Cs_smooth_sphere[:3], marker=".", facecolors='none', edgecolors="Blue", label="Standard data from clift", s=30, linewidths=0.5)

    ax1.set_xticks(x_ticks)
    ax1.set_yticks(y_ticks)
    ax1.minorticks_on()
    ax1.set_xlabel("Reynolds Number (Re)")
    ax1.set_ylabel("Drag Coefficient (Cd)")

    ax1.grid(which='major', linestyle='-', linewidth=0.75)
    ax1.grid(which='minor', linestyle=':', linewidth=0.5)
    ax1.legend(loc="lower right")

    # plots for 202mm sphere
    ax2.scatter(exp_Re[3:],  experimental_Cd_string_sphere[3:], color="green", marker="x", label="Measured sphere + string",  s=30, linewidths=0.5)
    ax2.scatter(exp_Re[3:], experimental_Cd_corrected_string_sphere[3:], marker="d", facecolors='none', edgecolors="goldenrod", label="string drag correction", s=30, linewidths=0.5)
    ax2.scatter(exp_Re[3:], experimental_Cd_solid_blockage[3:], marker="*",facecolors='none', edgecolors="magenta", label="Corrected for string drag and solid blockage", s=30, linewidths=0.5)
    ax2.scatter(exp_Re[3:], experimental_Cd_wake_string_sphere[3:], marker="^",facecolors='none', edgecolors="red", label="Corrected for string drag, solid and wake blockage", s=30, linewidths=0.5)
    ax2.scatter(exp_Re[3:], Cs_smooth_sphere[3:], marker=".", facecolors='none', edgecolors="Blue", label="Standard data from clift", s=30, linewidths=0.5)

    ax2.set_xticks(x_ticks)
    ax2.set_yticks(y_ticks)
    ax2.minorticks_on()
    ax2.set_xlabel("Reynolds Number (Re)")
    ax2.set_ylabel("Drag Coefficient (Cd)")

    ax2.grid(which='major', linestyle='-', linewidth=0.75)
    ax2.grid(which='minor', linestyle=':', linewidth=0.5)
    ax2.legend(loc="lower right")


    fig.text(0.10, 0.01, f"50mm sphere avg wind speeds: {velocities_50mm}                              202mm sphere average wind speeds: {velocities_202mm}", ha='left', fontsize=10)
    #fig.text(0.72, -0.02, f" 202mm sphere average wind speeds: {velocities_202mm}", ha='right', fontsize=10)

    plt.tight_layout(rect=[0, 0.03, 1, 1])  # Leave space at bottom for text
    plt.show()



```

```python
def main():
    """ main fuction call"""

    gas_constent = 287.6               # gas constant {JK^-1 kg^-1}
    speed_of_sound = 343.6          # speed of sound in meter per second

    sting_diameter = 0.01 # string diameter in meters
    string_frontal_area = 0.00215 # string frontal area {m^2}
    string_lenght = 0.00215   # string lenght in meters

    diameter_50mm = 0.05 # diameter 50mm sphere[m]
    diameter_202mm = 0.202 # diameter 202 sphere [m]

    frontal_area_50mm = 0.25 * np.pi* diameter_50mm ** 2    # frontal area in [m^2]
    frontal_area_202mm = 0.25 * np.pi* diameter_202mm ** 2  # frontal area in [m^2]

    tot_frontal_area_50mm = frontal_area_50mm + string_frontal_area
    tot_frontal_area_202mm = frontal_area_202mm + string_frontal_area

    temperature_conversion = 273.15 # conversion factor to get degrees to kelvin
    Temperature = 21        # temperature in degrees
    P_air = 101900          #aphmospheric air pressure
    kinematic_viscosity_air = 1.51*10**-5 # kinematic viscosity [m^2/s]

    air_density =  (P_air )/(gas_constent*(Temperature + temperature_conversion))



    """data for blockage correction"""
    n_power = 1
    k3s = 0.41
    H_tunnel = 0.905 # wind tunnel height [m]
    B_breadth = 1.22 # wind tunnel breadth [m]
    cross_sectional_area = 1.1041 # wind tunnel crossestion area [m^2]

    std_data = "emperical_data_clift.csv"

    data_set_1 = "50mm_25ms.txt"
    data_set_2 = "50mm_30ms.txt"
    data_set_3 = "50mm_40ms.txt"

    data_set_4 = "202mm_7.5ms.txt"
    data_set_5 = "202mm_10ms.txt"
    data_set_6 = "202mm_12.5ms.txt"

    data_set_7 = "sting_7.5ms.txt"
    data_set_8 = "sting_10ms.txt"
    data_set_9 = "sting_12.5ms.txt"

    data_set_10 = "sting_25ms.txt"
    data_set_11 = "sting_30ms.txt"
    data_set_12 = "sting_40ms.txt"

    files7_5ms = [data_set_4, data_set_7]
    files10ms = [data_set_5, data_set_8]
    files12_5ms = [data_set_6, data_set_9]

    files25ms = [data_set_1, data_set_10]
    files30ms = [data_set_2, data_set_11]
    files40ms = [data_set_3, data_set_12]

    clift_data = load_clift_data(std_data)


    # Collect experimental Re and Cd values
    experimental_Re = []

    experimental_Cd_string_sphere = []
    experimental_Cd_string_correction = []
    experimental_Cd_solid_blockage = []
    experimental_Cd_wake_string_sphere = []

    std_smooth_sphere = []
    ave_velocity_list = []
```

```python


    # Define a list of test cases
    test_cases = [
        (files7_5ms, diameter_202mm, frontal_area_202mm, tot_frontal_area_202mm),
        (files10ms, diameter_202mm, frontal_area_202mm, tot_frontal_area_202mm),
        (files12_5ms, diameter_202mm, frontal_area_202mm, tot_frontal_area_202mm),
        (files25ms, diameter_50mm, frontal_area_50mm, tot_frontal_area_50mm),
        (files30ms, diameter_50mm, frontal_area_50mm, tot_frontal_area_50mm),
        (files40ms, diameter_50mm, frontal_area_50mm, tot_frontal_area_50mm),
    ]

    for files, diameter, area, total_area in test_cases:
        # drag cofficients
        Cd_sphere_string, Cd_string, velocity_sphere = caculate_Cd(files[0], files[1], air_density, diameter, total_area, string_frontal_area)
        # reynalds numbers
        Re_sphere_string, _ = caculate_Reynalds_number(files[0], files[1], diameter, sting_diameter, kinematic_viscosity_air)
        # blockage correction
        _, _, _, Cd_sphere_corr_solid_blockage, _, _, cd_tot_wake_block = blockage_corrolation(
            files[0], files[1], speed_of_sound, diameter, area, n_power, k3s,
            H_tunnel, B_breadth, cross_sectional_area, air_density)
         # string drag correction
        string_drag = string_drag_corrolation(files[0], files[1], air_density, area)  # This gives corrected Cd
        Cd_string_drag_correction = string_drag[3]


        # Take average Re and Cd
        avg_Re = np.mean(Re_sphere_string)
        ave_Cd_string_sphere = np.mean(Cd_sphere_string)
        avg_Cd_string_drag_correction = np.mean(Cd_string_drag_correction)
        avg_Cd_solid_blockage = np.mean(Cd_sphere_corr_solid_blockage)
        avg_Cd = cd_tot_wake_block

        Cd_smooth_sphere = (24 / avg_Re) * (1+0.15*avg_Re ** 0.687) + (0.42 / (1+ 42500 * (avg_Re ** -1.16)))
        avg_velocity_sphere = np.sum(velocity_sphere) / np.size(velocity_sphere)


        #print(avg_Re)
        #print(avg_Cd)
        #print(Cs_smooth_sphere)

        experimental_Re.append(avg_Re)
        experimental_Cd_string_sphere.append(ave_Cd_string_sphere)
        experimental_Cd_solid_blockage.append(avg_Cd_solid_blockage)
        experimental_Cd_wake_string_sphere.append(avg_Cd)
        std_smooth_sphere.append(Cd_smooth_sphere)

        experimental_Cd_string_correction.append(avg_Cd_string_drag_correction)
        ave_velocity_list.append(round(avg_velocity_sphere,1))

        #print(ave_Cd_string_sphere)

    #print(experimental_Re)
    #print(experimental_Cd_string_sphere)
    #print(experimental_Cd_string_correction)
    #print(experimental_Cd_solid_blockage)
    #print(std_smooth_sphere)
    # print(ave_velocity_list)

    log_plot_Re_Cd(clift_data, air_density, experimental_Re, experimental_Cd_wake_string_sphere)

    linear_plot_Re_Cd(experimental_Re,
                      experimental_Cd_string_sphere,
                      experimental_Cd_string_correction,
                      experimental_Cd_solid_blockage,
                      experimental_Cd_wake_string_sphere,
                      std_smooth_sphere,
                      ave_velocity_list)



main()
```