Intro to Turbulent Flow
HW1

Almog Dobrescu ID 214254252

May 26, 2025

Contents

List of Figures

Listings

1 Computing a Turbulent Flow

Considering a commercial airliner with a chord $L = 5\,[\mathrm{m}]$ cruising at $U = 250\left[\frac{\mathrm{m}}{\mathrm{sec}}\right]$.

1.1 Estimating boundary layer thickness δ

According to Prandtl's one-seventh power law:

$$\frac{\delta}{x} \approx \frac{0.16}{Re_x^{\frac{1}{7}}} \tag{1}$$

At the trailing edge, the thickness of the boundary layer is:

$$\frac{\delta}{L} \approx \frac{0.16}{Re_L^{\frac{1}{7}}} \quad , \quad Re_L = \frac{UL}{\nu} \tag{2}$$

To calculate the kinematic viscosity of air, let's assume the airliner is cruising at 30[kft]. At this hight, the temperature is about $-44.3°[C]$ [1]. Hence, the kinematic viscosity is about $1 \cdot 10^{-5}$ [2]

$$Re_L = \frac{250 \cdot 5}{1 \cdot 10^{-5}} = 1.25 \cdot 10^8 \tag{3}$$

$$\Downarrow$$

$$\delta \approx \frac{0.16 \cdot 5}{(1 \cdot 10^{-5})^{\frac{1}{7}}} = 0.0558[\mathrm{m}] \tag{4}$$

1.2 Smallest Dynamically Important Scales

At the largest scales, the Reynolds number is very high, which indicates a turbulent regime. In a turbulent regime, there is no dissipation, which means:

$$Re_\ell = \frac{u\ell}{\nu} \gg 1 \tag{5}$$

However, we know that turbulent flows are dissipative, so there must be dissipation. For dissipation to accrue, the Reynolds number for the scales at which dissipation happens, the smallest scales, must be:

$$Re_\eta = \frac{v\eta}{\nu} \sim 1 \tag{6}$$

We can see that by fulfilling both demands for turbulent flow, there must be an energy transfer between scales, namely, an energy cascade.

1.3 Number of Grid Points

In order to accurately calculating the flow at the boundary layer, the size of one cell needs to be smaller then smallest eddy. From the energy cascade concept, we can conclude that the rate of change of the kinetic energy at the large scales:

$$\frac{du^2}{dt} \sim \frac{u^2}{\frac{\ell}{u}} \sim \varepsilon \tag{7}$$

is balanced by the energy dissipated at the small scales. By dimensional analysis, we find that:

$$\varepsilon \sim \nu \left(\frac{v}{\eta}\right)^2 \tag{8}$$

By combining Eq.7 and Eq.8 we get:

$$\frac{\ell}{\eta} \sim Re_\ell^{\frac{3}{4}} \quad \text{and} \quad \frac{v}{u} \sim Re_\ell^{-\frac{1}{4}} \quad \text{and} \quad \frac{\frac{\ell}{u}}{\frac{\eta}{v}} \sim Re_\ell^{\frac{1}{2}} \tag{9}$$

From Eq.9 we can calculate the size of one cell at the boundary layer:

$$\eta_x \sim \frac{L}{Re_L^{\frac{3}{4}}} = \frac{5}{(1.25 \cdot 10^8)^{\frac{3}{4}}} = 4.2295 \cdot 10^{-6}\,[\text{m}] \tag{10}$$

So, the number of grid point along the chord is:

$$N_x \sim \frac{L}{\eta} = \frac{5}{4.2295 \cdot 10^{-6}} = 1.1822 \cdot 10^6 \tag{11}$$

To calculate the number of grid point normal to the airfoil, we need to estimate the small scale of the boundary layer:

$$\eta_y \sim \frac{\delta}{Re_\delta{\frac{3}{4}}} = \frac{0.0558}{\left(\frac{U\delta}{\nu}\right)^{\frac{3}{4}}} = 1.3745 \cdot 10^{-6}\,[\text{m}] \tag{12}$$

so the number of grid point normal to the airfoil:

$$N_y \sim \frac{\delta}{\eta} = \frac{0.0558}{4.2295 \cdot 10^{-6}} = 4.0574 \cdot 10^4 \tag{13}$$

The total number of grid point is therefore:

$$N = N_x N_y = \boxed{4.7965 \cdot 10^{10}} \tag{14}$$

1.4 Instantaneous Flow Over One Eddy

The turnover time of one eddy is defined as:

$$t_{large} = \frac{\delta}{U} \tag{15}$$

The time step needs to be smaller then the smallest time step. Therefore:

$$t_{small} = \frac{\eta_y}{v} \tag{16}$$

From Eq.9 we get:

$$\frac{t_{large}}{t_{small}} \sim Re_\delta^{\frac{1}{2}} \tag{17}$$

$$\Downarrow$$

$$N_t = \frac{t_{large}}{t_{small}} \sim Re_\delta^{\frac{1}{2}} \sim 1180 \text{ steps} \tag{18}$$

I finished this quistion in about 3 hours

2 Heating a Room

2.1 Characteristic Time Scale

The one dimensional heat equation for a non moving field is:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{19}$$

We will use dimensional analysis to determined the characteristic time scale for heating the room.

$$\begin{aligned} \frac{T_\infty}{t} &= \alpha \frac{T_\infty}{L^2} \\ t &= \frac{L^2}{\alpha} \end{aligned} \tag{20}$$

Assuming typical bedroom conditions [3]:

$$\alpha|_{T\,=\,25^\circ [C]} = 22.39 \cdot 10^{-6} \left[\frac{\mathrm{m}^2}{\mathrm{sec}}\right] \quad , \quad L = 3\,[\mathrm{m}]$$

$$\Downarrow$$

$$t = 4.0197 \cdot 10^5\,[\mathrm{sec}] = 4.6524\,[\mathrm{day}] \tag{21}$$

2.2 Induced Velocity Estimation

The momentum equation under Boussinesq approximation can be written as:

$$\frac{D}{Dt}\vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + g\frac{\Delta T}{T} \tag{22}$$

Where:

- u is velocity induced by a space heater via buoyancy effects
- T is the local temperature
- ΔT is the difference between T and the newly heated air

Assuming the buoyancy-driven flow is turbulent:

$$Re \gg 1$$

The momentum equation can be rewritten as:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + g\frac{\Delta T}{T}$$

and by substituting the operators we get:

$$\frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu \frac{\partial^2 u}{\partial^2 x} + g\frac{T - T_h}{T} \tag{23}$$

In order to estimate the magnitude of the induced velocity we will use dimensional analysis:

$$u = u_h \tilde{u} \;\Big|\; t = T\tilde{t} \;\Big|\; x = h\tilde{x} \;\Big|\; p = \Lambda \tilde{p} \;\Big|\; T = T_\infty \tilde{T}$$

$$\frac{u_h}{T}\frac{\partial \tilde{u}}{\partial \tilde{t}} + \frac{u_h^2}{h}\tilde{u}\cdot\frac{\partial \tilde{u}}{\partial \tilde{x}} = -\frac{\Lambda}{\rho h}\frac{\partial \tilde{p}}{\partial \tilde{x}} + \frac{\nu u_h}{h^2}\frac{\partial^2 \tilde{u}}{\partial^2 \tilde{x}} + g\frac{\tilde{T} - \frac{T_h}{T_\infty}}{\tilde{T}} \tag{24}$$

Multiplying by $\dfrac{h^2}{\nu u_h}$:

$$\frac{h^2}{\nu u_h}\frac{u_h}{T}\frac{\partial \tilde{u}}{\partial \tilde{t}} + \frac{h^2}{\nu u_h}\frac{u_h^2}{h}\tilde{u}\cdot\frac{\partial \tilde{u}}{\partial \tilde{x}} = -\frac{h^2}{\nu u_h}\frac{\Lambda}{\rho h}\frac{\partial \tilde{p}}{\partial \tilde{x}} + \frac{h^2}{\nu u_h}\frac{\nu u_h}{h^2}\frac{\partial^2 \tilde{u}}{\partial^2 \tilde{x}} + \frac{h^2}{\nu u_h}g\frac{\tilde{T} - \frac{T_h}{T_\infty}}{\tilde{T}} \tag{25}$$

$$\underbrace{\frac{u_h h}{\nu}}_{Re_h}\underbrace{\frac{h}{u_h T}}_{St_h}\frac{\partial \tilde{u}}{\partial \tilde{t}} + \underbrace{\frac{h u_h}{\nu}}_{Re_h}\tilde{u}\cdot\frac{\partial \tilde{u}}{\partial \tilde{x}} = -\underbrace{\frac{u_h h}{\nu}}_{Re_h}\frac{\Lambda}{\rho u_h^2}\frac{\partial \tilde{p}}{\partial \tilde{x}} + \frac{\partial^2 \tilde{u}}{\partial^2 \tilde{x}} + \underbrace{\frac{u_h h}{\nu}}_{Re_h}\frac{h}{u_h^2}g\frac{\tilde{T} - \frac{T_h}{T_\infty}}{\tilde{T}} \tag{26}$$

For a buoyancy-driven turbulent flow, the dominant terms are the advection and buoyancy terms. Hence, we can estimate the following relations:

$$\begin{gathered}\frac{Re_h \Lambda}{\rho u_h^2} \sim 1 \quad , \quad \frac{Re_h h g}{u_h^2} \sim Re_h \\ \Downarrow \\ \Lambda \sim \frac{\rho u_h^2}{Re_h} \quad , \quad u_h \sim \sqrt{hg}\end{gathered} \tag{27}$$

Therefore, the magnitude of the induced velocity, for $h = 0.3[m]$ is:

$$u_h \sim \sqrt{0.3 \cdot 9.81} = 1.7155 \left[\frac{\mathrm{m}}{\mathrm{sec}}\right] \tag{28}$$

2.3 Characteristic Time Scale For Room Heating

Let's consider a cloud of particles in turbulent flow. For two particles in that cloud, if the the two particles spread by Fickian diffusion, then their spread can by described by the diffusivity constant D:

$$r^2 \sim Dt \tag{29}$$

From Richardson's 4/3's law, we now:

$$D_{\text{turb}} \sim \varepsilon^{\frac{1}{3}} r^{\frac{4}{3}} \tag{30}$$

Where $\varepsilon = \dfrac{u^3}{L}$ from the lecture.
Therefore, the characteristic time scale for heating the room is:

$$t \sim \frac{L^2}{\varepsilon^{\frac{1}{3}} L^{\frac{4}{3}}} = \frac{L^2}{\frac{u_h}{L^{\frac{1}{3}}} L^{\frac{4}{3}}} = \frac{L}{u_h} = 1.7487[sec] \tag{31}$$

I finished this quistion in one day

3 Help Karl Pearson

3.1 Number of Realizations

A Drunk walker takes n random steps. In our case, $n = 100$. In order to determine how many walks a walker need to take, in order for the PDF to converge, we will run tests for a range of realizations.
The PDF is calculated by counting all realizations inside a circle with radius r and a bigger circle with radius $r + dr$. In our case $dr = 0.5$:

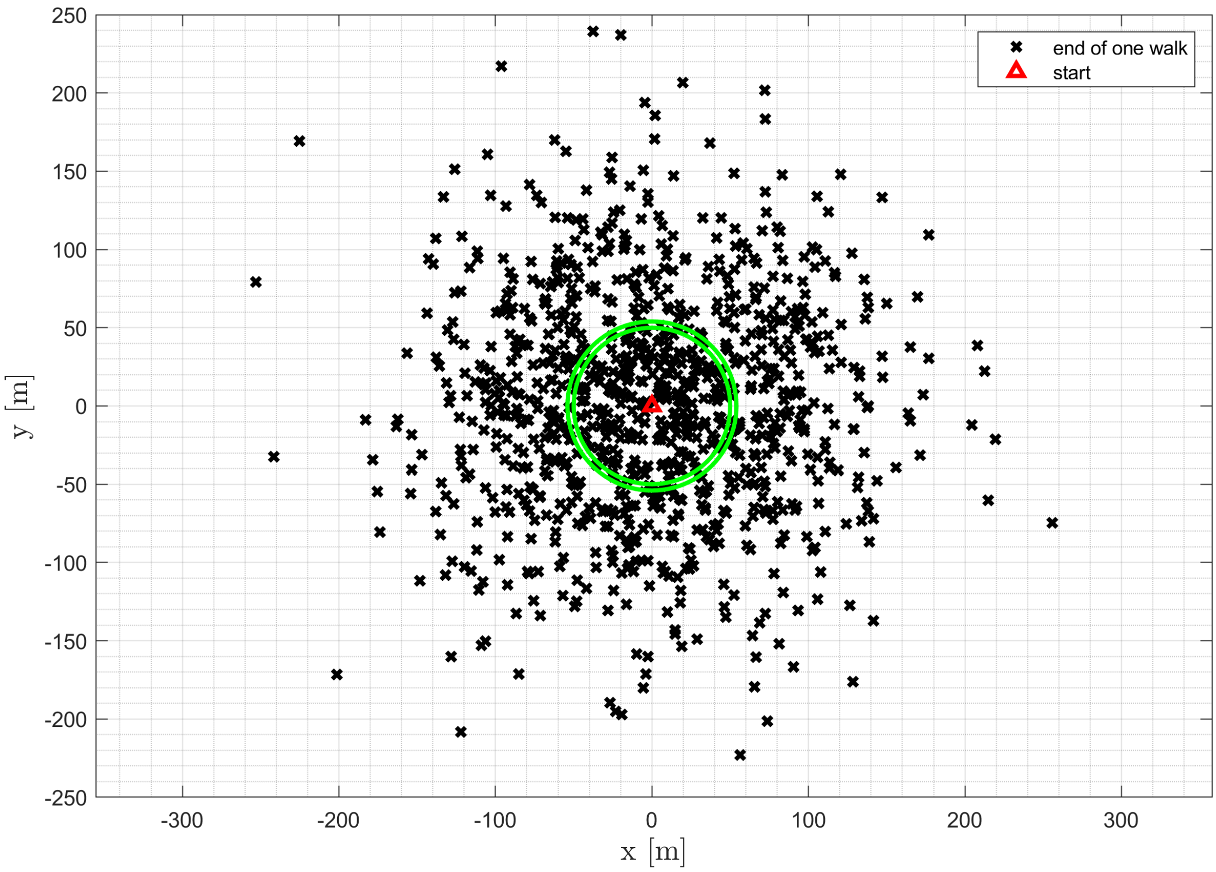

Figure 1: Calculating PDF

Lord Rayleigh's result for the ensemble PDF is:

$$\frac{2}{n} r e^{-\frac{r^2}{n}} \tag{32}$$

- for a step of $1[m]$

We calculated the PDF function as a function of r for different numbers of realizations and compared the difference between the calculated PDF and Rayleigh's equation. The RMS is calculated by:

$$RMS = \sqrt{\sum_{r=0}^{r_{max}} \left[PDF_{analytical}(r) - PDF_{numerical}(r)\right]^2} \tag{33}$$

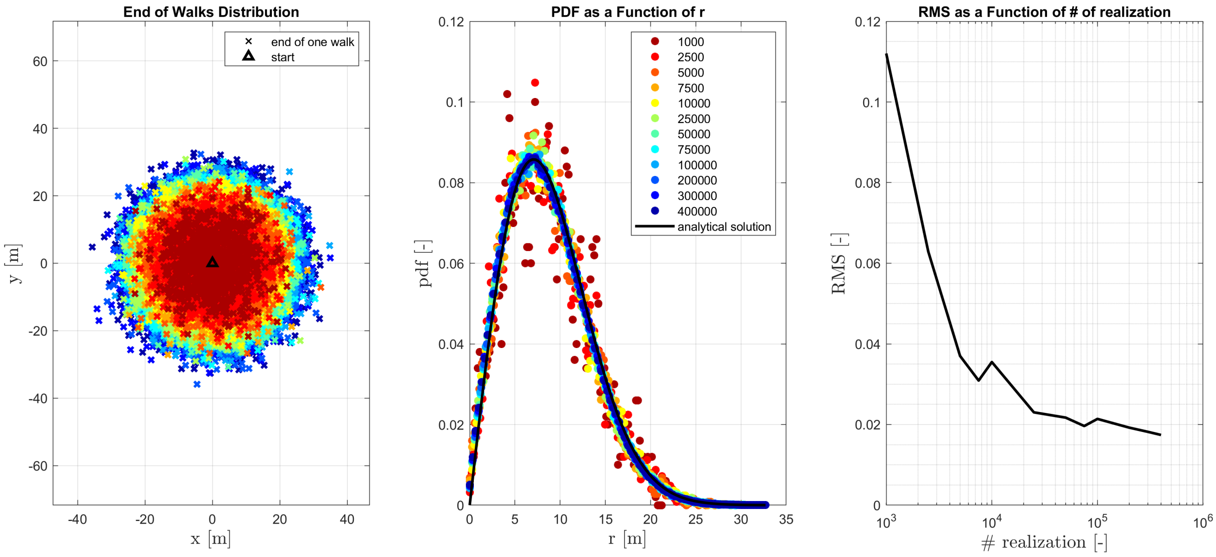

Figure 2: PDF function as a function of realizations

We can see that for number of realizations bigger then 10^5, we can see that the RMS between the calculated PDF and Rayleigh's equation is small enough to consider converged.

3.2 Average Displacement

From the lectures, the average displacement $\langle r \rangle$:

$$\langle r \rangle = \mathbb{E}(r) = \int_{-\infty}^{\infty} r \cdot PDF(r)\, dr \tag{34}$$

Hence, the average displacement as a function of number of realizations is:

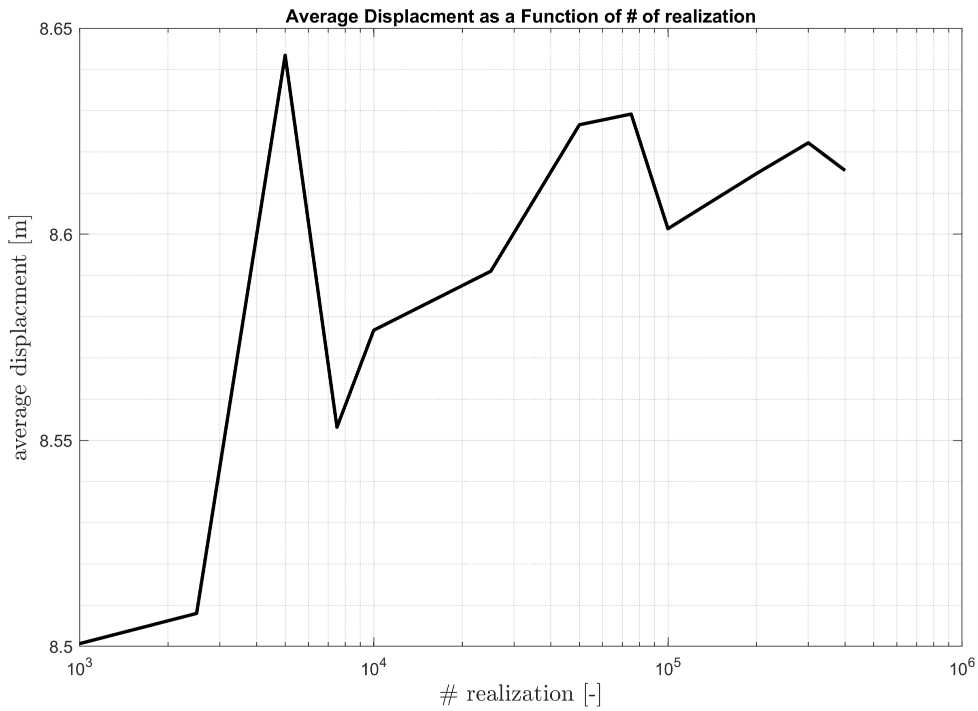

Figure 3: Average Displacement as a function of realizations

We can see that the average displacement converge to a value between 8.6 and 8.65.

3.3 Autocovariance and Autocorrelation

The covariance is calculated using the *cov* function in *MatLab*. The autocorrelation is the normalized autocovariance. The definition of the covariance is:

$$Q_{uv} = \mathbb{E}\left[(u - \mu_u)(v - \mu_v)\right] \tag{35}$$

The autocovariance is therefore:

$$Q(\Delta\ell) = \mathbb{E}\left[(r(\ell) - \mu)(r(\ell + \Delta\ell) - \mu)\right] \tag{36}$$

and the autocorrelation is:

$$R(\Delta\ell) = \frac{Q(\Delta\ell)}{\max\{Q(\Delta\ell)\}} \tag{37}$$

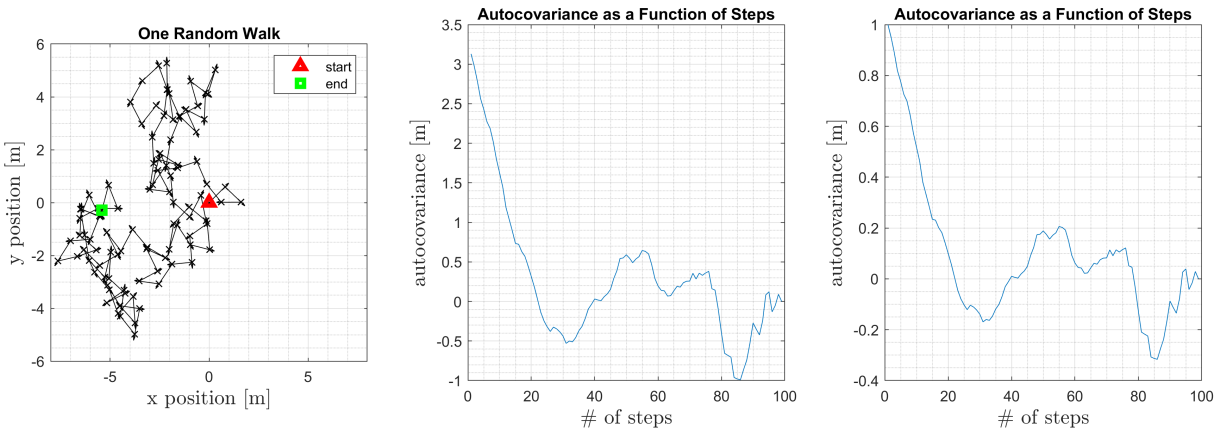

Figure 4: Autocovariance and Autocorrelation

In order to determine how many walks are needed in the ensemble for the autocorrelation to converge we will calculate the autocorrelation over a range of realizations:

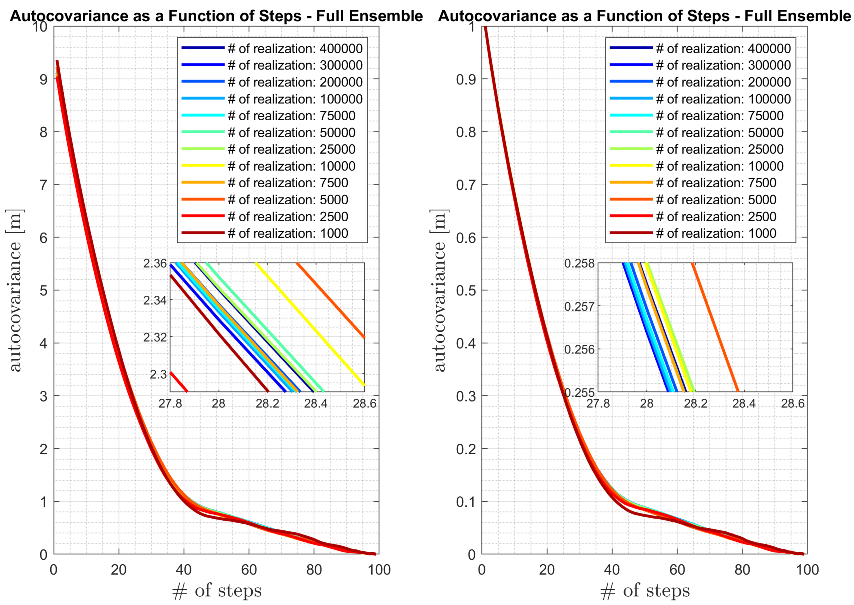

Figure 5: Autocovariance and Autocorrelation ensemble

We can see that for more that 10^4 realizations, the autocovariance does not have any significant changes.
I finished this quistion in 3 day

References

[1] T. E. ToolBox, "U.s. standard atmosphere: Temperature, pressure, and air properties vs. altitude." https://www.engineeringtoolbox.com/standard-atmosphere-d_604.html, 2003.

[2] E. Edge, "Viscosity of air, dynamic and kinematic." https://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm, 2025.

[3] T. E. ToolBox, "Air - thermal diffusivity vs. temperature and pressure." https://www.engineeringtoolbox.com/air-thermal-diffusivity-d_2011.html, 2018.

A Listing of The Computer Program

A.1 Q1

```
clc; clear; close all

U      = 250;     % [m/sec]
L      = 5;       % [m]
nu_30k = 1e-5;    % [m^2/sec]

Re_L     = U * L / nu_30k
delta    = 0.16 * 5 / Re_L^(1/7)
Re_delta = U * delta / nu_30k
eta_x    = 5 / Re_L^(3/4)
eta_y    = delta / Re_delta^(3/4)
N_x      = L / eta_x
N_y      = delta / eta_y
N        = N_x * N_y
N_t      = Re_delta^0.5
```

Listing 1: Code for Q1

A.2 Q2

```
clc; clear; close all

L         = 3;               % [m]
h         = 0.3;             % [m]
g         = 9.81;            % [m/sec^2]
alpha_25C = 22.39e-6;        % [m^2/sec]
t_sec     = L^2/alpha_25C    % [sec]
t_day     = t_sec/60/60/24   % [day]

u_h       = sqrt(h*g)        % [m/sec]
t_heating = L/u_h            % [sec]
```

Listing 2: Code for Q2

A.3 Q3

```
clc; clear; close all;

%% Q.3.a PREFORM AND PRINT A RUN ########################################
step_size = 1; % [m]
num_of_steps = 1e4;
num_of_walks = 1e3;
r  = 50;
dr = 4;

fig1 = figure('Name', '1', 'Position', [100, 250, 900, 600]);
hold all

steps = [];
[end_x_vec, end_y_vec, ~] = one_run(0, 0, step_size, num_of_steps, num_of_walks);

[count_of_band, count_of_outer_circ] = num_points_in_band_and_outer_circ(r, dr, end_x_vec
    , end_y_vec);
```

```matlab

plot(end_x_vec, end_y_vec, 'x', 'LineWidth', 2, 'Color', 'k')
plot(0,0, '^', 'LineWidth', 2, 'Color', 'r')
draw_circ(0, 0, r, 'g', 2)
draw_circ(0, 0, r+dr, 'g', 2)

ax = gca;
MaxX = max(abs(ax.XLim));
MaxY = max(abs(ax.YLim));
axis([-MaxX MaxX -MaxY MaxY]);

xlabel('x [m]','FontSize',14,'Interpreter','latex')
ylabel('y [m]','FontSize',14,'Interpreter','latex')
legend({'end of one walk','start'})
axis equal
box on
grid on
grid minor
% exportgraphics(fig1, 'images/graph1.png','Resolution',300);

%% Q.3.a PREFORM AND PRINT A RUN AND PDF ################################
step_size = 1; % [m]
num_of_steps     = 1e2;
dr               = 0.5;
num_of_walks_vec = [4e5, 3e5, 2e5, 1e5, 7.5e4, 5e4, 2.5e4, 1e4, 7.5e3, 5e3, 2.5e3, 1e3];
% num_of_walks_vec = [7.5e4, 5e4, 2.5e4, 1e4, 5e3, 1e3];
num_of_radiuses  = 1e2;

end_x_vec_vec        = {};
end_y_vec_vec        = {};
distances_vec_of_vec = {};
rs_vec               = {};
pdf_vec_vec          = {};
for index = 1:length(num_of_walks_vec)
    num_of_walks = num_of_walks_vec(index);
    steps = [];
    [current_end_x_vec, current_end_y_vec, current_distances_vec] = one_run(0, 0,
        step_size, num_of_steps, num_of_walks);
    end_x_vec_vec{index,1} = current_end_x_vec;
    end_y_vec_vec{index,1} = current_end_y_vec;
    distances_vec_of_vec{index,1} = current_distances_vec;
    rs_vec{index,1} = linspace(0, min(max(abs(current_end_x_vec)),max(abs(
        current_end_y_vec))), num_of_radiuses);
    pdf_vec_vec{index,1} = calc_pdf(rs_vec{index,1}, dr, current_end_x_vec,
        current_end_y_vec);
end

fig2 = figure('Name', '2', 'Position', [150, 250, 1500, 600]);
hold all
colors = jet(length(num_of_walks_vec));
% colors = cool(length(num_of_walks_vec))*0.8;
lg  = {};
rms = [];
for index = 1:length(num_of_walks_vec)
    num_of_walks = num_of_walks_vec(index);
    lg{end+1} = sprintf('%g', double(num_of_walks_vec(end-index+1)));
    end_x_vec = end_x_vec_vec{index,1};
    end_y_vec = end_y_vec_vec{index,1};
```

```

    subplot(1,3,1) % ####################################################
    hold all
    p = plot(0,0, 'x', 'LineWidth', 1, 'Color', 'k', 'HandleVisibility', 'callback');
    plot(end_x_vec, end_y_vec, 'x', 'LineWidth', 2, 'Color', colors(index,:))
    s = plot(0,0, '^', 'LineWidth', 2, 'Color', 'k');

    ax = gca;
    MaxX = max(abs(ax.XLim));
    MaxY = max(abs(ax.YLim));
    axis([-MaxX MaxX -MaxY MaxY]);
    xlabel('x [m]','FontSize',14,'Interpreter','latex')
    ylabel('y [m]','FontSize',14,'Interpreter','latex')
    title('End of Walks Distribution')
    legend([p, s], {'end of one walk','start'})
    axis equal
    % axis square
    box on
    grid on
    grid minor

    subplot(1,3,2) % ####################################################
    hold all

    rs                  = rs_vec{length(num_of_walks_vec)-index+1, 1};
    analytical_solution = 2/num_of_steps.*rs.*exp(-rs.^2/num_of_steps);
    pdf_vec             = pdf_vec_vec{length(num_of_walks_vec)-index+1, 1};

    plot(rs, pdf_vec, '*', 'LineWidth', 2, 'Color', colors(length(num_of_walks_vec)-index+1,:))
    if index == length(num_of_walks_vec)
        plot(rs, analytical_solution, '-', 'LineWidth', 2, 'Color', 'k')
        lg{end+1} = 'analytical solution';
    end

    xlabel('r [m]','FontSize',14,'Interpreter','latex')
    ylabel('pdf [-]','FontSize',14,'Interpreter','latex')
    title('PDF as a Function of r')
    legend(lg)
    box on
    grid on
    grid minor

    subplot(1,3,3) % ####################################################

    rms(end+1) = sqrt(sum((analytical_solution-pdf_vec).^2));

    semilogx(flip(num_of_walks_vec(end-length(rms)+1:end)), rms, '-', 'LineWidth', 2, 'Color', 'k')

    xlabel('\# realization [-]','FontSize',14,'Interpreter','latex')
    ylabel('RMS [-]','FontSize',14,'Interpreter','latex')
    title('RMS as a Function of # of realization')
    box on
    grid on
    grid minor
end
% exportgraphics(fig2, 'images/graph2.png','Resolution',300);
```

```
% exportgraphics(fig1, 'images/graph1.png','Resolution',300); exportgraphics(fig2, '
    images/graph2.png','Resolution',300);

%% Q.3.b Average Displacement ###########################################
% step_size = 1; % [m]
% num_of_steps     = 1e2;
% dr               = 0.5;
% % num_of_walks_vec = [4e5, 3e5, 2e5, 1e5, 7.5e4, 5e4, 2.5e4, 1e4, 7.5e3, 5e3, 2.5e3, 1
    e3];
% num_of_walks_vec = [7.55e4, 5e4, 2.5e4, 1e4, 5e3, 1e3];
% num_of_radiuses  = 1e2;
%
% end_x_vec_vec        = {};
% end_y_vec_vec        = {};
% distances_vec_of_vec = {};
% rs_vec               = {};
% pdf_vec_vec          = {};
% for index = 1:length(num_of_walks_vec)
%     num_of_walks = num_of_walks_vec(index);
%     steps = [];
%     [current_end_x_vec, current_end_y_vec, current_distances_vec] = one_run(0, 0,
    step_size, num_of_steps, num_of_walks);
%     end_x_vec_vec{index,1} = current_end_x_vec;
%     end_y_vec_vec{index,1} = current_end_y_vec;
%     distances_vec_of_vec{index,1} = current_distances_vec;
%     rs_vec{index,1} = linspace(0, min(max(abs(current_end_x_vec)),max(abs(
    current_end_y_vec))), num_of_radiuses);
%     pdf_vec_vec{index,1} = calc_pdf(rs_vec{index,1}, dr, current_end_x_vec,
    current_end_y_vec);
% end

ave_disp = [];
for index = 1:length(num_of_walks_vec)
    ave_disp(index) = trapz(rs_vec{index,:}, rs_vec{index,:}.*pdf_vec_vec{index,:});
end

fig3 = figure('Name', '3', 'Position', [300, 250, 900, 600]);

semilogx(num_of_walks_vec, ave_disp, '-', 'LineWidth', 2, 'Color', 'k')

xlabel('\# realization [-]','FontSize',14,'Interpreter','latex')
ylabel('average displacment [m]','FontSize',14,'Interpreter','latex')
title('Average Displacment as a Function of # of realization')
box on
grid on
grid minor
% exportgraphics(fig3, 'images/graph3.png','Resolution',300);
% exportgraphics(fig3, 'images/graph3.png','Resolution',300); exportgraphics(fig1, '
    images/graph1.png','Resolution',300); exportgraphics(fig2, 'images/graph2.png','
    Resolution',300);

%% Q.3.c autocovariance and autocorrelation ############################
step_size = 1; % [m]
number_of_steps = 1e2;

fig4 = figure('Name', '4', 'Position', [450, 250, 1350, 400]);

subplot(1,3,1) % #######################################################
```

```
hold all
steps = [];
distances = [];
steps(1,:) = [0,0];
for i=1:number_of_steps
    [x,y] = one_step(steps(i, 1), steps(i, 2), step_size);
    steps(end+1,:) = [x,y];
    distances(end+1) = sqrt(x^2+y^2);
end

plot(steps(:,1), steps(:,2), 'x', 'LineWidth', 0.75, 'Color', 'k', 'HandleVisibility','off')
quiver(steps(1:end-1,1), steps(1:end-1,2), diff(steps(:,1)), diff(steps(:,2)), 'MarkerSize', 0.5, 'Color', 'k', 'MaxHeadSize', 0.05, 'HandleVisibility','off')
plot(steps(1,1),steps(1,2), '^', 'LineWidth', 3, 'Color', 'r')
plot(steps(end,1),steps(end,2), 'squar', 'LineWidth', 3, 'Color', 'g')

ax = gca;
MaxX = max(abs(ax.XLim));
MaxY = max(abs(ax.YLim));
axis([-MaxX MaxX -MaxY MaxY]);

xlabel('x position [m]','FontSize',14,'Interpreter','latex')
ylabel('y position [m]','FontSize',14,'Interpreter','latex')
title('One Random Walk')
legend({'start','end'})
axis square
box on
grid on
grid minor

Q = [];
for i = 1:length(distances)
    A = distances(1:length(distances)-i);
    B = distances(1+i:length(distances));
    C = cov(A, B);
    Q(i) = C(1,2);
end

subplot(1,3,2) % ########################################################
plot(1:length(distances), Q)

xlabel('\# of steps','FontSize',14,'Interpreter','latex')
ylabel('autocovariance [m]','FontSize',14,'Interpreter','latex')
title('Autocovariance as a Function of Steps')
box on
grid on
grid minor

subplot(1,3,3) % ########################################################
plot(1:length(distances), Q/Q(1))

xlabel('\# of steps','FontSize',14,'Interpreter','latex')
ylabel('autocovariance [m]','FontSize',14,'Interpreter','latex')
title('Autocovariance as a Function of Steps')
box on
grid on
```

```
grid minor
% exportgraphics(fig4, 'images/graph4.png','Resolution',300);
% exportgraphics(fig4, 'images/graph4.png','Resolution',300); exportgraphics(fig3, 'images/graph3.png','Resolution',300); exportgraphics(fig1, 'images/graph1.png','Resolution',300); exportgraphics(fig2, 'images/graph2.png','Resolution',300);

%% Q.3.c ensemble ######################################################
% step_size = 1; % [m]
% num_of_steps     = 1e2;
% dr               = 0.5;
% % num_of_walks_vec = [4e5, 3e5, 2e5, 1e5, 7.5e4, 5e4, 2.5e4, 1e4, 7.5e3, 5e3, 2.5e3, 1e3];
% num_of_walks_vec = [7.55e4, 5e4, 2.5e4, 1e4, 5e3, 1e3];
% num_of_radiuses  = 1e2;
%
% end_x_vec_vec        = {};
% end_y_vec_vec        = {};
% distances_vec_of_vec = {};
% rs_vec               = {};
% pdf_vec_vec          = {};
% for index = 1:length(num_of_walks_vec)
%     num_of_walks = num_of_walks_vec(index);
%     steps = [];
%     [current_end_x_vec, current_end_y_vec, current_distances_vec] = one_run(0, 0, step_size, num_of_steps, num_of_walks);
%     end_x_vec_vec{index,1} = current_end_x_vec;
%     end_y_vec_vec{index,1} = current_end_y_vec;
%     distances_vec_of_vec{index,1} = current_distances_vec;
%     rs_vec{index,1} = linspace(0, min(max(abs(current_end_x_vec)),max(abs(current_end_y_vec))), num_of_radiuses);
%     pdf_vec_vec{index,1} = calc_pdf(rs_vec{index,1}, dr, current_end_x_vec, current_end_y_vec);
% end

Q_vec = {};
Q_ensemble = {};
for index = 1:length(num_of_walks_vec)
    fprintf('Walk number %d\n', index);
    distances_vec = distances_vec_of_vec{index,1};
    Q_ensemble{index,1} = zeros(1,length(distances));
    for j = 1:length(distances_vec)
        j
        distances = distances_vec{j,1};
        Q = [];
        for i = 1:length(distances)
            A = distances(1:length(distances)-i);
            B = distances(1+i:length(distances));
            C = cov(A, B);
            Q(i) = C(1,2);
        end
        Q_ensemble{index,1} = Q_ensemble{index,1} + Q;
    end
    Q_ensemble{index,1} = Q_ensemble{index,1} / num_of_walks_vec(index);
end
%%
fig5 = figure('Name', '5', 'Position', [600, 250, 900, 600]);

colors = jet(length(num_of_walks_vec));
```

```
% colors = cool(length(num_of_walks_vec))*0.8;
subplot(1,2,1) % ########################################################
hold all
lg  = {};
for index = 1:length(num_of_walks_vec)
    Q = Q_ensemble{index,1};
    plot(1:length(distances), Q, '–', 'LineWidth', 2, 'Color', colors(index,:))
    lg{end+1} = sprintf('# of realization: %d', num_of_walks_vec(index));
end

xlabel('\# of steps','FontSize',14,'Interpreter','latex')
ylabel('autocovariance [m]','FontSize',14,'Interpreter','latex')
title('Autocovariance as a Function of Steps – Full Ensemble')
legend(lg)
box on
grid on
grid minor

zoom = axes('position',[0.25 0.36 0.2 0.2]);
box on % put box around new pair of axes
hold all
for index = 1:length(num_of_walks_vec)
    Q = Q_ensemble{index,1};
    plot(1:length(distances), Q, '–', 'LineWidth', 2, 'Color', colors(index,:))
end
zoom.XLim = [27.8, 28.6];
zoom.YLim = [2.29, 2.36];
grid on
grid minor

subplot(1,2,2) % ########################################################
hold all
lg  = {};
for index = 1:length(num_of_walks_vec)
    Q = Q_ensemble{index,1};
    plot(1:length(distances), Q/Q(1), '–', 'LineWidth', 2, 'Color', colors(index,:))
    hold on
    lg{end+1} = sprintf('# of realization: %d', num_of_walks_vec(index));
end

xlabel('\# of steps','FontSize',14,'Interpreter','latex')
ylabel('autocovariance [m]','FontSize',14,'Interpreter','latex')
title('Autocovariance as a Function of Steps – Full Ensemble')
legend(lg)
box on
grid on
grid minor

zoom = axes('position',[0.69 0.36 0.2 0.2]);
box on % put box around new pair of axes
hold all
for index = 1:length(num_of_walks_vec)
    Q = Q_ensemble{index,1};
    plot(1:length(distances), Q/Q(1), '–', 'LineWidth', 2, 'Color', colors(index,:))
end
zoom.XLim = [27.8, 28.6];
zoom.YLim = [0.255, 0.258];
grid on
```

```
grid minor

% exportgraphics(fig5, 'images/graph5.png','Resolution',300);
% exportgraphics(fig5, 'images/graph5.png','Resolution',300); exportgraphics(fig4, 'images/graph4.png','Resolution',300); exportgraphics(fig3, 'images/graph3.png','Resolution',300); exportgraphics(fig1, 'images/graph1.png','Resolution',300); exportgraphics(fig2, 'images/graph2.png','Resolution',300);








% FUNCTIONS ############################################################
function [des_x, des_y] = one_step(src_x, src_y, step_size)
    theta = rand()*2*pi;
    des_x = src_x + step_size * cos(theta);
    des_y = src_y + step_size * sin(theta);
end

function [end_x, end_y, distances] = one_walk(start_x, start_y, step_size, num_of_steps)
    x = start_x;
    y = start_y;
    distances = [];
    for i=1:num_of_steps
        [x,y] = one_step(x, y, step_size);
        distances(i) = sqrt(x^2+y^2);
    end
    end_x = x;
    end_y = y;
end

function [end_x_vec, end_y_vec, distances_vec] = one_run(start_x, start_y, step_size, num_of_steps, num_of_walks)
    fprintf('preforming a run ...\n');
    steps = [];
    distances_vec = {};
    for i=1:num_of_walks
        if ~mod(i, num_of_walks/10)
            fprintf('    complited: %2.0f%%\n', i/num_of_walks*100);
        end
        [x,y,distances] = one_walk(start_x, start_y, step_size, num_of_steps);
        steps(end+1,:) = [x,y];
        distances_vec{i, 1} = distances;
    end
    end_x_vec = steps(:,1);
    end_y_vec = steps(:,2);
end


function [count_of_band, count_of_outer_circ] = num_points_in_band_and_outer_circ(r, dr, x_vec, y_vec)
    if length(x_vec) ~= length(y_vec)
        fprintf('x and y are not the same length\n');
        return
    end

```

```
    count_of_inner_circ = 0;
    count_of_outer_circ = 0;
    for i = 1:length(x_vec)
        dist_squre = x_vec(i)^2+y_vec(i)^2;
        if dist_squre <= r^2
            count_of_inner_circ = count_of_inner_circ+1;
        end
        if dist_squre <= (r+dr)^2
            count_of_outer_circ = count_of_outer_circ+1;
        end
    end

    count_of_band = count_of_outer_circ - count_of_inner_circ;
end

function draw_circ(center_x, center_y, r, color, line_width)
    pos = [[center_x, center_y]-r, 2*r, 2*r];
    rectangle('Position',pos,'Curvature',[1 1], 'EdgeColor', color, 'LineWidth', line_width)
end

function pdf_vec = calc_pdf(rs, dr, x_vec, y_vec)
    if length(x_vec) ~= length(y_vec)
        fprintf('x and y are not the same length\n');
        return
    end
    fprintf('calc pdf, dr = %4f\n', dr);

    pdf_vec = [];
    for i=1:length(rs)
        r = rs(i);
        [count_of_band, count_of_outer_circ] = num_points_in_band_and_outer_circ(r, dr, x_vec, y_vec);
        cdf_inner = (count_of_outer_circ-count_of_band) / length(x_vec); % points inside circ
        cdf_outer = (count_of_outer_circ) / length(x_vec); % points outside circ
        pdf_vec(i) = (cdf_outer - cdf_inner) / dr;
    end
end
```

Listing 3: Code for Q3