Table of Contents

PREAMBLE

```
close all;
clear variables;
clc;

set(groot, 'defaultTextInterpreter', 'Latex');
set(groot, 'defaultLegendInterpreter', 'Latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
set(groot, 'defaultTextFontSize', 12);
set(groot, 'defaultAxesFontSize', 16);
set(groot, 'defaultLineLineWidth', 2);
set(groot, 'defaultFigureColor', 'white');
```

Loading Data

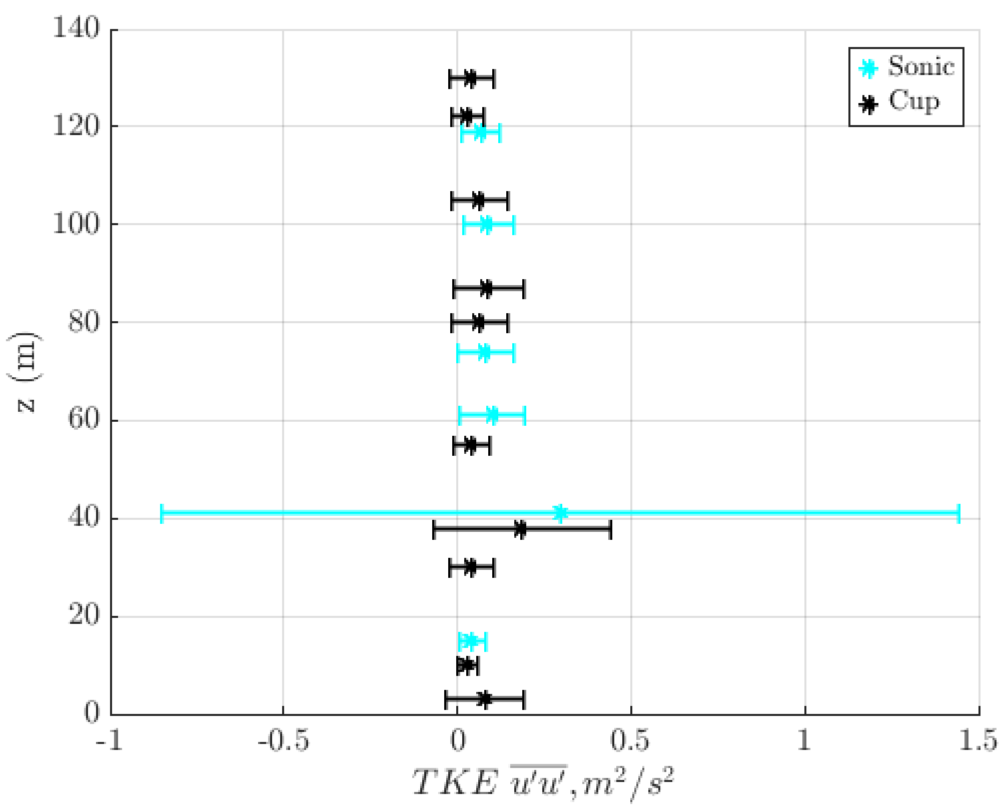
```
%REMEMBER TO CHANGE THIS IF ON WINDOWS cause apples hates me
fileDir = '/Users/christopherbianco/Desktop/School_Code/Wind Physics/HW1';
%Mac

%fileDir = 'C:\Users\Christopher\Desktop\School_Code\Wind Physics\HW1';
%Windows

data = load(fullfile(fileDir, '08_28_2019_10_00_00_000.mat'));
```

Part a: TKE Measurements

```
%Initialize measurement height
sonic_heights = [15,41,61, 74, 100, 119];
U_av_sonic = NaN(1, length(sonic_heights));
U_std_sonic = NaN(1, length(sonic_heights));
TKE_sonic = NaN(1, length(sonic_heights));
TKE_std_sonic = NaN(1, length(sonic_heights));


cup_heights = [3, 10, 30, 38, 55, 80, 87, 105, 122, 130];
```

```
U_av_cup = NaN(1, length(cup_heights));
U_std_cup = NaN(1, length(cup_heights));
TKE_cup = NaN(1, length(cup_heights));
TKE_std_cup = NaN(1, length(cup_heights));


%Calculate horizontal wind speed from sonic anemometers
for i = 1:length(sonic_heights)
    sonic_height = sonic_heights(i);
    U = sqrt(data.(strcat('Sonic_x_clean_',num2str(sonic_height),'m')).val.^2
+ data.(strcat('Sonic_y_clean_',num2str(sonic_height),'m')).val.^2);

    U_av_sonic(i) = mean(U);
    U_std_sonic(i) = std(U);

    u = data.(strcat('Sonic_x_clean_',num2str(sonic_height),'m')).val;
    v = data.(strcat('Sonic_y_clean_',num2str(sonic_height),'m')).val;
    w = data.(strcat('Sonic_z_clean_',num2str(sonic_height),'m')).val;


    %TKE
    up = u - mean(u);
    vp = v - mean(v);
    wp = w - mean(w);

    TKE_sonic(i) = (1/2)*(mean(up.^2) + mean(vp.^2) + mean(wp.^2));
    TKE_std_sonic(i) = std((1/2)*(up.^2 + vp.^2 + wp.^2));

end

%Extract horizontal wind speed from cup anemometers
for i = 1:length(cup_heights)
    cup_height = cup_heights(i);

    %Now, we need to take into account the naming convention
    fn = fieldnames(data);
    matchIdx = contains(fn, 'Cup_WS_') & contains(fn,
strcat(num2str(cup_height),'m'));
    match = fn(matchIdx);

    %Extract U
    U = data.(match{1}).val;

    %Do mean and standard deviation
    U_av_cup(i) = mean(U);
    U_std_cup(i) = std(U);

    up = U - mean(U);
    TKE_cup(i) = mean(up.^2);
    TKE_std_cup(i) = std(up.^2);
end

%Make the figure
figure(1); hold on;
```

```
xlabel('$TKE ~\overline{u^{\prime}u^{\prime}}, m^2/s^2$');
ylabel('z (m)');
grid on

%Plot sonic
e1 = errorbar(TKE_sonic, sonic_heights, TKE_std_sonic, 'horizontal', '*c',
'LineWidth',2);
%Plot cup
e2 = errorbar(TKE_cup, cup_heights, TKE_std_cup, 'horizontal', '*k',
'LineWidth',2);

% Dummy plots for legend only
h1 = plot(nan, nan, '*c', 'LineWidth', 2);
h2 = plot(nan, nan, '*k', 'LineWidth', 2);

% Legend
legend([h1 h2], {'Sonic','Cup'}, 'Location', 'best')

hold off
```



Part a short answer:

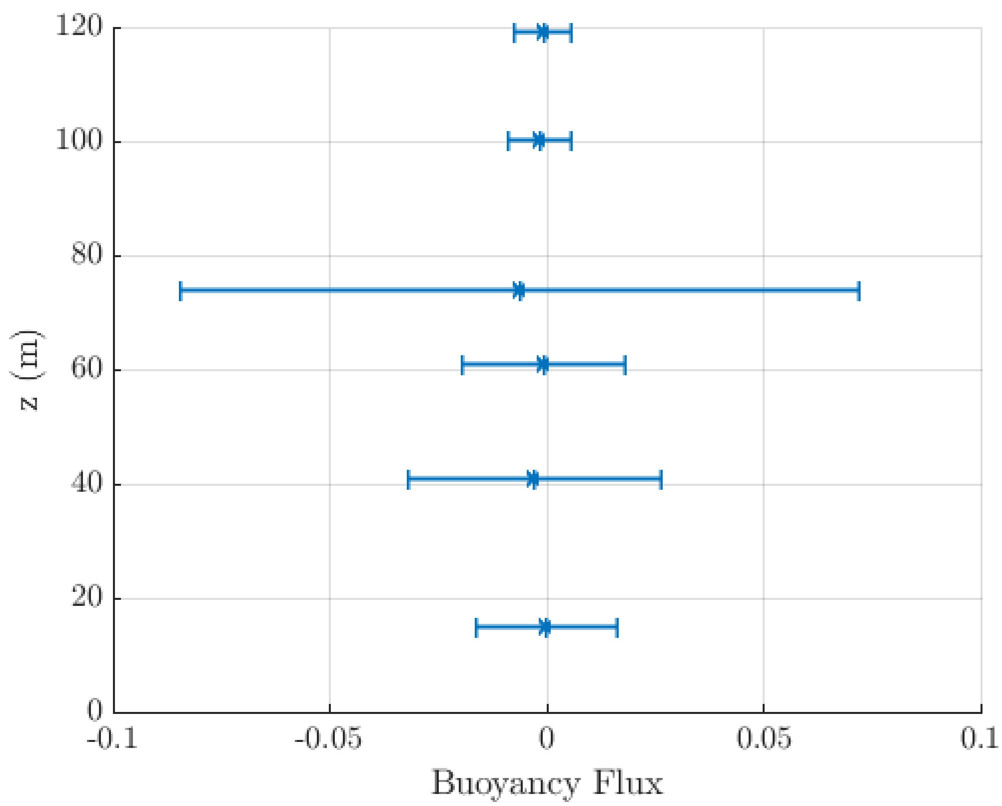
```
%The general trend of TKE follows figure 5.1 in Stull (staying close to
%zero and barely increasing with height). We do see very high standard
%deviation.
```

Part b: Buoyancy Flux

Assuming the mixing ratio is small, we can say that the virtual temperature is just equal to the temperature and calculate virtual potential temperature (VPT) from there. Essentially, we're neglecting the humidity correction as it is likely small.

```
%Initialize measurement height
sonic_heights = [15,41,61,74,100,119];
temp_av_sonic = NaN(1, length(sonic_heights));
temp_std_sonic = NaN(1, length(sonic_heights));
bf_av_sonic =  NaN(1, length(sonic_heights));
bf_std_sonic =  NaN(1, length(sonic_heights));

% I don't think we can get horizontal wind speed for the solo measurements
% solo_heights = [3, 38, 87];
% temp_av_solo = NaN(1, length(solo_heights));
% temp_std_solo = NaN(1, length(solo_heights));
% bf_av_solo = NaN(1, length(solo_heights));
% bf_std_solo = NaN(1, length(solo_heights));
%
% cup_heights = [3,38, 87];


%Extract temp from sonic anemometers
for i = 1:length(sonic_heights)
    sonic_height = sonic_heights(i);
    temp = data.(strcat('Sonic_Temp_clean_', num2str(sonic_height), 'm')).val;

    %Get average temp
    temp_av_sonic(i) = mean(temp);
    temp_std_sonic(i) = std(temp);

    %Extract vertical velocity
    w = data.(strcat('Sonic_z_clean_',num2str(sonic_height),'m')).val;

    %Calculate pressure
    p = 100*mean(data.Baro_Presr_3m.val) +
sonic_height.*9.81.*100*mean(data.Baro_Presr_3m.val)./(287.05.*(mean(temp)));

    %Calculate vpt
    vpt = temp.*(81100/p)^(287/1004);

    %Calculate buoyancy flux
    bf = (9.81/mean(vpt)).*((w - mean(w)).*(vpt - mean(vpt)));
    bf_av_sonic(i) = mean(bf);
    bf_std_sonic(i) = std(bf);

end

% Again, don't think we can get
% %Extract temp from stand alone measurements
% for i = 1:length(solo_heights)
%     solo_height = solo_heights(i);
```

```
%     temp = data.(strcat('Air_Temp_', num2str(solo_height), 'm')).val;
%
%     temp_av_solo(i) = mean(temp);
%     temp_std_solo(i) = std(temp);
% end

%Make figure
figure(2); hold on;
xlabel('Buoyancy Flux');
ylabel('z (m)');
grid on

%Plot sonic
e1 = errorbar(bf_av_sonic, sonic_heights, bf_std_sonic, 'horizontal', '*',
'LineWidth',2);
%Plot cup
%e2 = errorbar(temp_av_solo, solo_heights, temp_std_solo, 'horizontal', '*b',
'LineWidth',2);

% Dummy plots for legend only
h1 = plot(nan, nan, '*c', 'LineWidth', 2);
h2 = plot(nan, nan, '*k', 'LineWidth', 2);
hold off
```



Problem b short answer

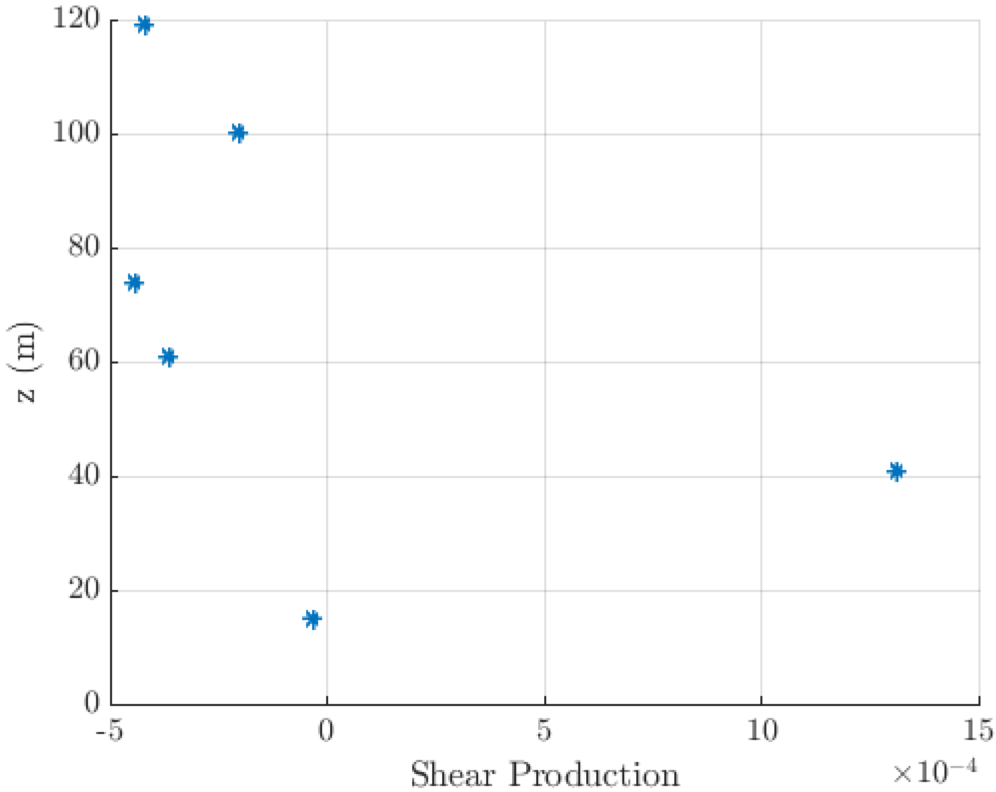
```
%As can be seen from the plot, we have almost no buoyancy production, with
%a slight negative trend.
%This corresponds to stable stratification, which makes sense
%for 4am local time.
```

Problem c

Problem c


```
%Initialize measurement height
sonic_heights = [15,41,61, 74, 100, 119];
shear_prod = NaN(1, length(sonic_heights));

%Calculate horizontal wind speed from sonic anemometers
for i = 1:length(sonic_heights)
    sonic_height = sonic_heights(i);

    %Mean horizontal flow
    U = sqrt(data.(strcat('Sonic_x_clean_',num2str(sonic_height),'m')).val.^2
+ data.(strcat('Sonic_y_clean_',num2str(sonic_height),'m')).val.^2);
    w = data.(strcat('Sonic_z_clean_',num2str(sonic_height),'m')).val;


    %Fluctuations
    Up = U - mean(U);
    wp = w - mean(w);

    %dU/dz using finite differences

    if i == 1
        dU = (U_av_sonic(i+1) + U_av_sonic(i))/(sonic_heights(i+1)-
sonic_heights(i));
    elseif i == 6
        dU = (U_av_sonic(i) + U_av_sonic(i-1))/(sonic_heights(i)-
sonic_heights(i-1));
    else
        dU = (U_av_sonic(i+1) + U_av_sonic(i-1))/(sonic_heights(i+1)-
sonic_heights(i-1));
    end

    %Shear production
    shear_prod(i) = dU*mean(Up.*wp);
end

figure(3); hold on;
xlabel('Shear Production');
ylabel('z (m)');
grid on
```

```
plot(shear_prod, sonic_heights, '*', 'LineWidth',2);
hold off
```



Problem c short answer

```
%The magnitude of bouyant production is larger than that of shear
%production, and the buoyant production is slightly negative.
%According to Stull, this puts the flow in the stably stratified range.
```

Published with MATLAB® R2024b