University Oldenburg

Wind Physics Measurement Project

Exercise 1 - Handling and preprocessing of measurement data

Author:
Jan Kämper
Florian Börgel

Supervisor:
Mathias Wächter

May 3, 2016

Contents

Introduction

The goal of this exercise was to perform some basic processing steps of raw measurement data from a met mast. The data used here originates from the FINO 1 platform in the German Northsea which includes two wind vanes at heights of $33m$ and $90m$ as well as eight anemometers at heights $33m, 40m, 50m, 60m, 70m, 80m, 90m$ and $100m$. The given time period of 1-Hertz data is of one month (January 2013). The six tasks described in the following reach from standard data treatment to a first simple analysis by looking at the increment probability density function in terms of wind speed fluctuations.

1 Importing Data into Matlab

The first step was loading the data given as ASCII file into Matlab-readable data structures. For the first task we used the function *readtable()* to import the data with the corresponding delimiter as parameter. The advantage of using a table structure instead of a matrix is that the column headers like 'u100' or 'Time' are stored within the data structure. In this way we can easily separate the actual measurment data from the time stamp by splitting the table up in two matrices.

```
time_stamp = raw_data{:, {'Time'}};
raw_data = raw_data{:, {'d90', 'd33', 'u100', 'u90', 'u80', ...
'u70', 'u60', 'u50', 'u40', 'u33'}};
```

2 Marking invalid data

In order to mark invalid data which is provided with a value of -999 by the measurement system the next section of our Matlab script converts all values -999 to NaN. Matlab checks if there is any invalid Data and replaces it with NaN. This is necessary for some remaining tasks when means and standard deviations will be computed which must not consider invalid values.

```
raw_data(raw_data==-999) = NaN;
```

3 Generating a continuous time axis

To avoid gaps in the time axis we first converted our time t with *datenum()* to a numeric value. The numeric values represent elapsed time in units of days. Hence, 1 second corresponds to the fraction of $\frac{1}{24 \cdot 60 \cdot 60}$ of these numeric values. So after multiplication with the inverse value of that and rounding we obtain unique integer IDs for every occuring timestamp. Afterwards, we created the continuous time axis, by initializing a vector with length equaling exactly the number of seconds in January. Then, we filled the corrected data matrix *data_pp* with NaN values and overwrote the file with our existing data at all indices where values are given.

```
disp('Creating continous time axis')
tnew=[t(1):1:t(end)]';
data_pp = NaN(length(tnew),10);
disp('Writing preprocessed Data...')
for i = 1:length(raw_data(:,1))
    data_pp(t(i)-t(1)+1, :) = raw_data(i, :);
end
time = (1:length(data_pp))';
data_pp = [time, data_pp];
save('data_pp.mat', 'data_pp', 'raw_data');
clear;
```

4 Computing 10min means and standard deviation

This task is a first step of statistical analysis of the given data. We split our time axis into intervals of 600 seconds and for each interval we computed the mean and standard deviation for all ten variables. Invalid data can be ignored by using the commands *nanmean()* and *nanstd()*. Considering the wind directions a special treatment of the angles is required in order to handle the circular data, e.g. the mean value of 350° and 10° is not 180° but 0°. Some trigonometric functions can be employed in order to cope with this. We plotted the ten minute means of the $u90$ anemometer for one specific day and added the standard deviation to it. The outcome is depicted in figure 1.

For this specific day we observe a very strong fluctuation of the wind over the day. During the early morning the wind is steadily strong with windspeeds greater then $20m/s$ then drops by almost 60% untill 8am, then picks up again to reach a maximum of around $24m/s$ in the evening. Indeed a very stormy day. The mean value and standard deviation are obivously not stationary because

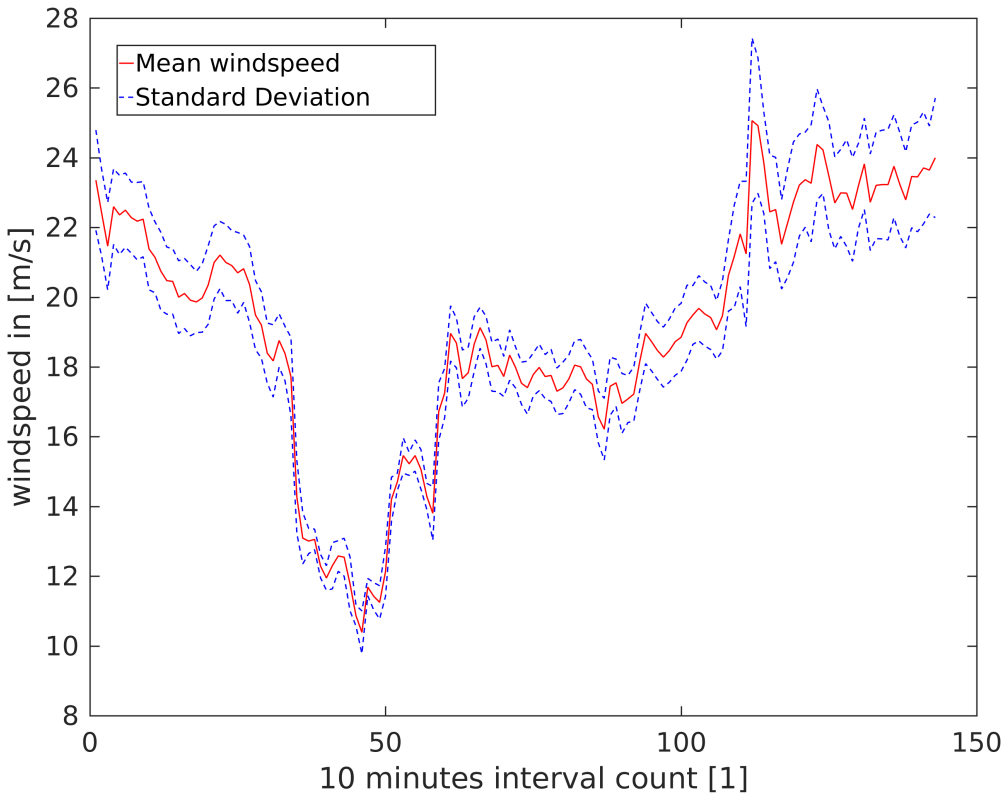

Figure 1: 10 minute intervals for Jan 30th 2013

then mean is high in the morning and evening and low inbetween. The fluctuations and thus the standard deviation picks up in the evening and therefore is also not stationary.

5 Extreme values

In this task we looked for so called spikes. These are extremely high or low values within their immediate neighborhood, here 10 minutes intervals. A standard procedure to define spikes is to find values which exceed $[\mu-5\sigma, \mu+5\sigma]$ in these intervals where μ is the mean and σ is the standard deviation.

Two such intervals and contained spikes where found by our algorithm and are plotted in Figures 2 and 3. The first shows a spike which appears to be of unphysical nature due to a relatively monotonous behaviour with only small fluctuations except one single value. This single value is below the $\mu - 5\sigma$ line. So here we could talk of a measurement error. The second plot however

exhibits a more physical behaviour. Here we have stronger fluctuations overall and the spike which is below $\mu - 5\sigma$ matches the fluctuational pattern because the values before and after the spike also deviate a lot from the mean over a considerable period of time.

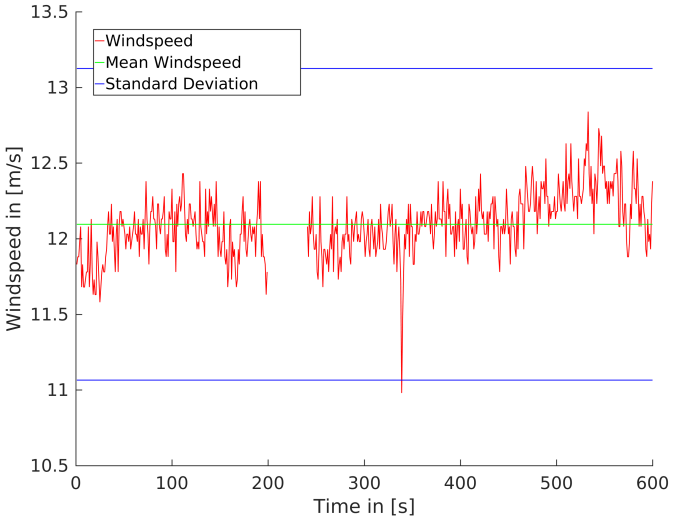

Figure 2: Unphysical cause

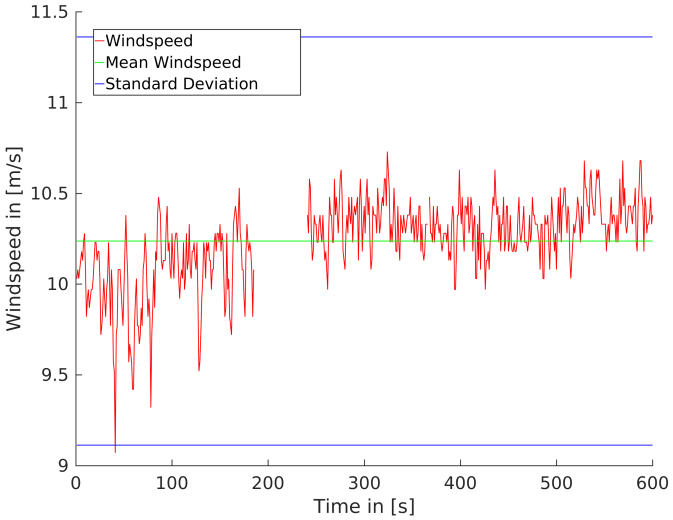

Figure 3: Physical cause

All in all we can say that the 5σ criterion is only an estimation of unusable values and relies on experience. By fixing the threshold to 5σ one states that all fluctuation within this interval are statistically significant. It implies reliability of the measurement system and is vulnerable to environmental influences, e.g. weather conditions.

6 Increment PDF

In this last task the aim was to compute the increment PDF. This probability density function defines the distribution of short term changes on the time axis. The formalism is $\delta u_\tau = u(t+\tau) - u(t)$ for a time lag of $\tau = 1s$. We collected the data in a histogram with 60 bins which gives a relatively smooth curve and normalized the values to unit standard deviation. We note that on this semi-

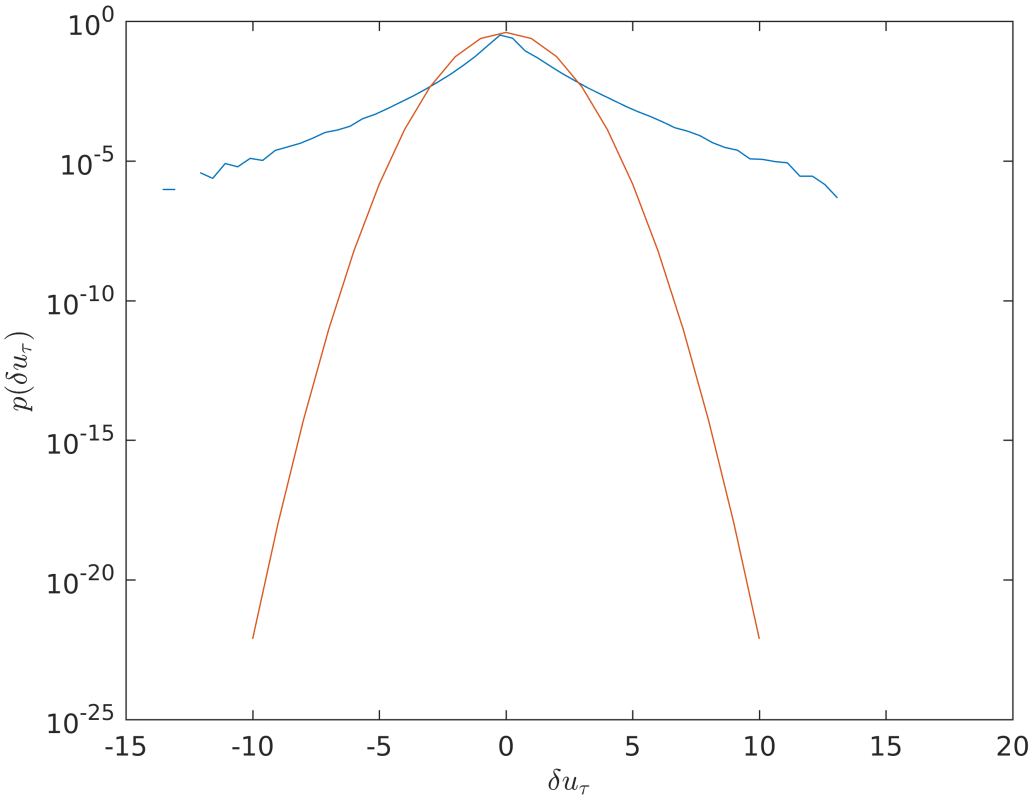

Figure 4: Increment PDF for $\tau = 1$ and Normal Distribution

logarithmic scale the relative frequencies of the $1sec$-increments diverge clearly from the normal distribution. Extreme values for the 1s fluctuation are much more likely then a normal distribution

suggests. In this case for increments of extreme values $\delta u = \pm 10 m/s$ we observe a difference up to an order of magnitude 10^{17} between the probabilities according to Gauss and the actual relative frequencies derived from the measurement data. For small absolute values of δu the Normal Distribution overestimates the acutal frequencies, but only to a maximal order of magnitude of 10^{1}. Considering the IEC standards the assumption of a Gaussian PDF is not conservative and does not take into account the high probability of extreme fluctuations. This should be considered by industry when designing turbines regarding load analysis.

A Appendix

```
%% preprocess data
disp('Preprocessing Data')
disp('Loading Data ...')
raw_data = readtable('1301.txt','Delimiter','tab');
time_stamp = raw_data{:, {'Time'}};
raw_data = raw_data{:, {'d90', 'd33', 'u100', 'u90', 'u80', ...
'u70', 'u60', 'u50', 'u40', 'u33'}};
t = (datenum(time_stamp, 'yyyy-mm-dd HH:MM:SS'));
clear time_stamp;
t = round(t*24*3600);
raw_data(raw_data==-999) = NaN;
last_time = 31*24*3600;
tnew=[1:1:last_time]';
n = length(tnew);
data_pp = NaN(length(tnew),10);
for i = 1:length(raw_data(:,1))
    data_pp(t(i)-t(1)+1, :) = raw_data(i, :);
end
time = (1:length(data_pp))';
data_pp = [time, data_pp];
save('data_pp.mat', 'data_pp', 'raw_data');
clearvars;
%% Evaluation
load('data_pp.mat');

% means and 10 min means
disp('Computing 10min means and stddev');
means_interval10 = NaN((length(data_pp(:,1))/600),20);
for i = 1:length(data_pp(:,1))/600
   %treat wind vanes
    radians = data_pp((i-1)*600+1:i*600,2)/180*pi;
    meanSin = nanmean(sin(radians));
    meanCos = nanmean(cos(radians));
    tanVal = atan2(meanSin,meanCos);
    means_interval10(i,1) = tanVal*180/pi;

    radiansPrime = radians - tanVal;
    primeUnwrapped = unwrap(radiansPrime);
    unwrappedStddev = nanstd(primeUnwrapped);
    means_interval10(i,2) = unwrappedStddev*180/pi;

    radians=data_pp((i-1)*600+1:i*600,3)/180*pi;
    meanSin = nanmean(sin(radians));
    meanCos = nanmean(cos(radians));
```

```
    tanVal = atan2(meanSin,meanCos);
    means_interval10(i,3) = tanVal*180/pi;

    radiansPrime = radians - tanVal;
    primeUnwrapped = unwrap(radiansPrime);
    unwrappedStddev = nanstd(primeUnwrapped);
    means_interval10(i,4) = unwrappedStddev*180/pi;
    %treat windspeeds u
    for j=3:10
        means_interval10(i,j*2-1) = nanmean(data_pp((i-1)*600+1:i*600,j+1));
        means_interval10(i,j*2) = nanstd(data_pp((i-1)*600+1:i*600,j+1));
    end
end

save('meansAndStddev.mat', 'means_interval10');
%% plot for January 30th
load('meansAndStddev.mat');
start30thJan = 29*24*6;
figure();
print -r1500
plot(means_interval10(start30thJan+1:start30thJan+24*6,7), '-r');
hold on;
plot(means_interval10(start30thJan+1:start30thJan+24*6,7)+means_interval10(
    start30thJan+1:start30thJan+24*6,8), '--b');
plot(means_interval10(start30thJan+1:start30thJan+24*6,7)-means_interval10(
    start30thJan+1:start30thJan+24*6,8), '--b');
xlabel('10 minutes interval count [1]');
ylabel('windspeed in [m/s]');
legend('Mean windspeed','Standard Deviation','Location','northwest');
disp('saving plot to Plots/mean_interval_withstd.png');
print('Plots/mean_interval_withstd.png','-dpng','-r500')
hold off;
clearvars;
%% spikes
load('meansAndStddev.mat');
load('data_pp.mat');
j= 1;
k = 1;
for i = 1:length(data_pp(:,1))/600
    if max((data_pp((i-1)*600+1:i*600,4))) > (means_interval10(i,5)+5*
    means_interval10(i,6))
        spikes(j,1) = i
        j = j+1;
    end
    if min((data_pp((i-1)*600+1:i*600,4))) < (means_interval10(i,5)-5*
    means_interval10(i,6))
        spikes(k,2) = i
```

```
            k = k+1;
89      end
    end
91
    % Plot spikes
93  i = 1084;
    window = 1
95  figure();
    for i = spikes(1:5,1)'
97      figure();
        plot_data_mean(1:600,1) = means_interval10(i,5);
99      plot_data_mean(1:600,2) = means_interval10(i,5)+5*means_interval10(i,6);
        plot_data_mean(1:600,3) = means_interval10(i,5)-5*means_interval10(i,6);
101
        hold on;
103     plot((data_pp((i-1)*600+1:i*600,4)), '-r')
        plot(plot_data_mean(1:600,1),'-g')
105     plot(plot_data_mean(1:600,2),'-b')
        plot(plot_data_mean(1:600,3),'-b')
107     xlabel('Time in [s]')
        ylabel('Windspeed in [m/s]')
109     legend('Windspeed','Mean Windspeed','Standard Deviation', 'Location', 'northwest
        ');
        hold off;
111     stri = num2str(i);
        str = strcat('Plots/spikesintervall', stri, '.png');
113     print(str,'-dpng','-r500');
    end
115 disp('saving plot to Plots/10min_interval_with_highspikes.png')
    saveas(gcf,'Plots/10min_interval_with_highspikes.png')
117
    window = 1
119 figure();
    for i = spikes(:,2)'
121     figure();
        plot_data_mean(1:600,1) = means_interval10(i,5);
123     plot_data_mean(1:600,2) = means_interval10(i,5)+5*means_interval10(i,6);
        plot_data_mean(1:600,3) = means_interval10(i,5)-5*means_interval10(i,6);
125
        hold on;
127     plot((data_pp((i-1)*600+1:i*600,4)), '-r')
        plot(plot_data_mean(1:600,1),'-g')
129     plot(plot_data_mean(1:600,2),'-b')
        plot(plot_data_mean(1:600,3),'-b')
131     xlabel('Time in [s]')
        ylabel('Windspeed in [m/s]')
```

```
133     legend('Windspeed','Mean Windspeed','Standard Deviation', 'Location', 'northwest
        ');
        hold off;
135     stri = num2str(i);
        str = strcat('Plots/spikesintervall', stri, '.png');
137     print(str,'-dpng','-r500');
    end
139 %disp('saving plot to Plots/10min_interval_with_lowerspikes.png')
    %saveas(gcf,'Plots/10min_interval_with_lowerspikes.png')
141 clearvars
    %% Task 6
143 load('data_pp.mat');
    load('meansAndStddev.mat');
145 tau = 1;
    for i = 1:length(data_pp(:,4))-1
147     du(i,1) = data_pp(i+tau,4)-data_pp(i,4);
    end
149

    du = du/nanstd(du);
151 [hist_y, hist_x] = hist(du,60);
    hist_y = hist_y/sum(hist_y);
153 gausx = -10:1:10;
    gausy = (1/sqrt(2*pi))*exp(-gausx.^2/2);
155

    figure();
157 semilogy(hist_x, hist_y);
    hold on;
159 semilogy(gausx,gausy);
    hold off;
161 set(0,'DefaultTextInterpreter', 'latex');
    xlabel('$\delta u_\tau$');
163 ylabel('$p(\delta u_\tau)$');
    print('Plots/tau_pdf_gauss10.png','-dpng','-r500')
165 clearvars
```