THE UNIVERSITY OF MELBOURNE

MCEN90018

Advanced Fluid Dynamics

Department of Mechanical Engineering

Assignment 3

Students:

Mischka Kamener, 539 030

Robert Haberkern, 637 517

Version: Final Version, 29/05/2016

Part 1 - Signal Analysis

All of the MATLAB code used to complete this section is given in Appendix A

Question 1

A low pass filter was implemented in MATLAB, and the code is shown below:

```
% MCEN90018:   Advanced Fluid Dynamics – Assignment 3
% ------------------------------------------------------------------------
% Mischka       Kamener     539030                    Last modified: 28/5/16
% Robert        Haberkern   637517
%
% Low pass filter that takes a signal of N samples with a total length of T
% seconds, and the cutoff frequency in Hz.

function filt = lpFilter(signal,T,N,cutoff)

% Get Fourier transform, and find cutoff index
g = fft(signal);
deltaf = 1/T;
f = ([0:1:N/2 (N/2-1):-1:1])'*deltaf; % Frequencies of each coefficient

% Set frequencies lower than cutoff to zero, and reverse transform
g(f>cutoff) = 0;
filt = real(ifft(g));
```

This filter was implemented on the hotfilm signal at $Y = 20$ for a cut-off of $50Hz$, and the unfiltered and filtered signal are plotted below.

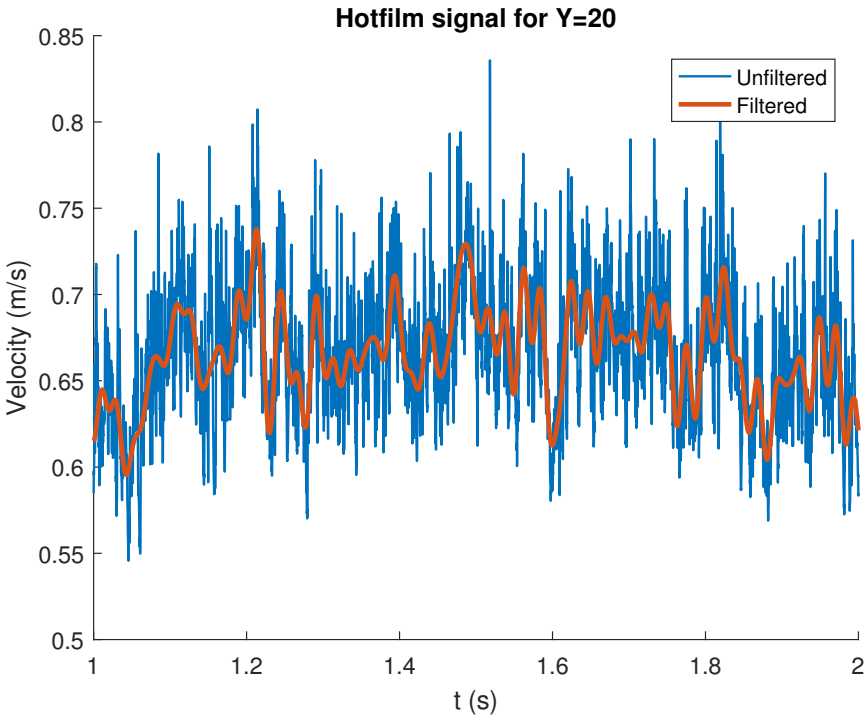

Figure 1: Low pass filtered signal with $50Hz$ cut-off

Question 2

The cross correlation of the hotfilm and hotwire sensor at $Y = 20$ was calculated using the long hand approach. Taylor's hypothesis was then used to convert the time shift into a displacement shift. The code used, and the correlation plot are given below. The hotfilm sensor was filtered before the correlation was calculated as per Question 1, as the frequencies greater than $50Hz$ are unreliable.

```
% Read file
f20 = readBin('f',20);
w20 = readBin('w',20);

f20 = lpFilter(f20,T,N,50);

% Find mean
Uc  = mean(w20);

% Adjust for mean and standard deviation
f20 = f20 - mean(f20); f20 = f20./std(f20);
w20 = w20 - mean(w20); w20 = w20./std(w20);

% Zero pad vector to be shifted
w20_p = [zeros(size(w20)); w20; zeros(size(w20))];
R_cc  = zeros(size(-N:N-1))';

tic
% Compute cross correlation. Slide one signal across the other
for n = -N:N-1;
    R_cc(n+N+1) = sum(f20.*w20_p((N+1:2*N)+n));
end
toc

% Normalise by number of elements
R_cc = R_cc./N;

% Determine dt values to plot
lim = 3*delta/Uc;
dt = (-N:N-1)./Fs;
dx = -dt*Uc;

% Plot graph
plot(dx(abs(dt)<lim)/delta,R_cc(abs(dt)<lim))
xlabel('\Deltax/\delta');
ylabel('R');
```

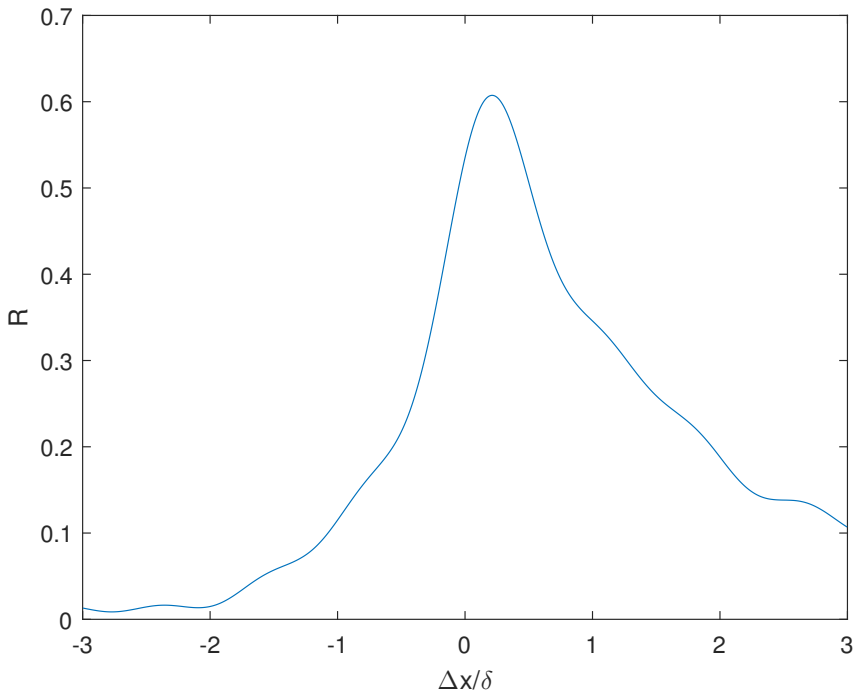

Figure 2: Long-hand cross correlation between hotfilm and hotwire sensor

The program execution was timed using `tic` and `toc`, and the correlation took 61.3 minutes to complete.

Since the sample positions were logarithmically spaced, the sample at $Y = 20$ corresponds to when the hotwire sensor was $8.593mm$ from the wall. The correlation peak between the two sensors occurs at a displacement of $\Delta x/\delta = 0.2124$, or $\Delta x = 69.2mm$. Since we know the positions of the two sensors that creates the greatest correlation, we can determine the angle at which the turbulent structures leave the wall.

$$\text{Structural inclination angle} = \arctan\left(\frac{8.593mm}{69.2mm}\right) = 7.074^{\circ} \tag{1.2.1}$$

Question 3

The same process was applied as in Question 2, however the correlation was instead calculated using the Fourier transform. The code used, and the correlation plot (from both methods) are shown below.

```
% Read files
f20 = readBin('f',20);
w20 = readBin('w',20);

f20 = lpFilter(f20,T,N,50);

% Find mean
Uc  = mean(w20);

% Adjust for mean and standard deviation
f20 = f20 - mean(f20); f20 = f20./std(f20);
w20 = w20 - mean(w20); w20 = w20./std(w20);

% Pad with zeros
```

```
f20_p = [zeros(size(f20)); f20];
w20_p = [zeros(size(w20)); w20];

tic
% Compute cross correlation using fft (R_fft goes from -N to N-1)
R_fft = fftshift(ifft(conj(fft(f20_p)).*(fft(w20_p))));
toc

% Normalise by number of elements
R_fft = R_fft./N;

% Determine dt values to plot
lim = 3*delta/Uc;
dt = (-N:N-1)./Fs;
dx = -dt*Uc;

% Plot graph
plot(dx(abs(dt)<lim)/delta,R_fft(abs(dt)<lim),'-', ...
    dx(abs(dt)<lim)/delta,R_cc(abs(dt)<lim),'--')
xlabel('\Deltax/\delta');
ylabel('R');
legend('FFT method', 'Long-hand method');
```

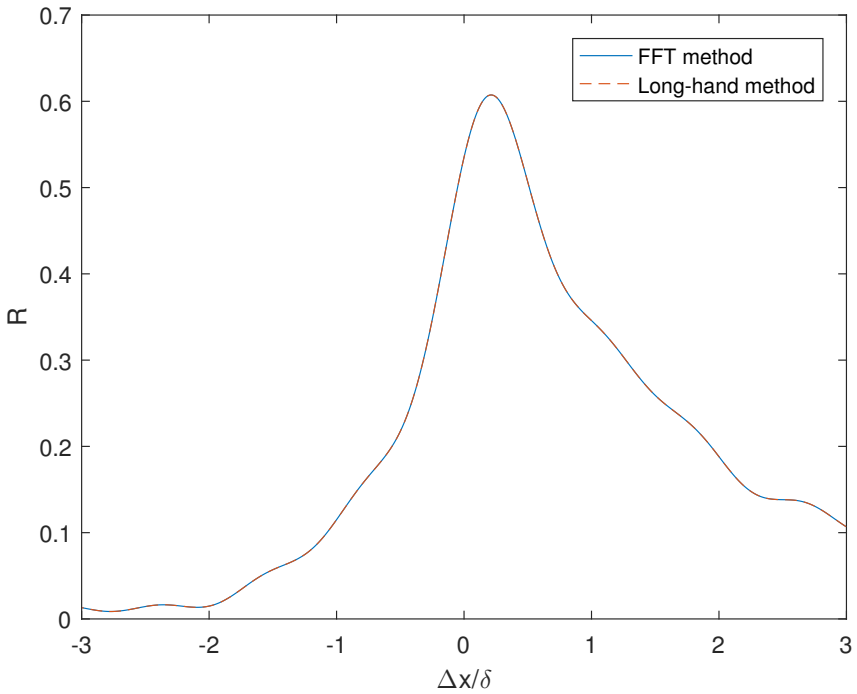

Figure 3: Cross correlation between hotfilm and hotwire sensor

The figure above shows that the two methods used produce identical results. The program execution was again timed using `tic` and `toc`, and the Fourier transform method took only 0.046427 seconds. This is about 80000 times faster than the long-hand approach, so the Fourier transform is by far the superior method.

Question 4

The cross correlation of all the Y positions was calculated, and Taylor's hypothesis was used to convert all of the time shifts to displacement shifts for the different mean velocities. The isocontours of R are plotted below. The code used to generate the plot is given in Appendix A.3.

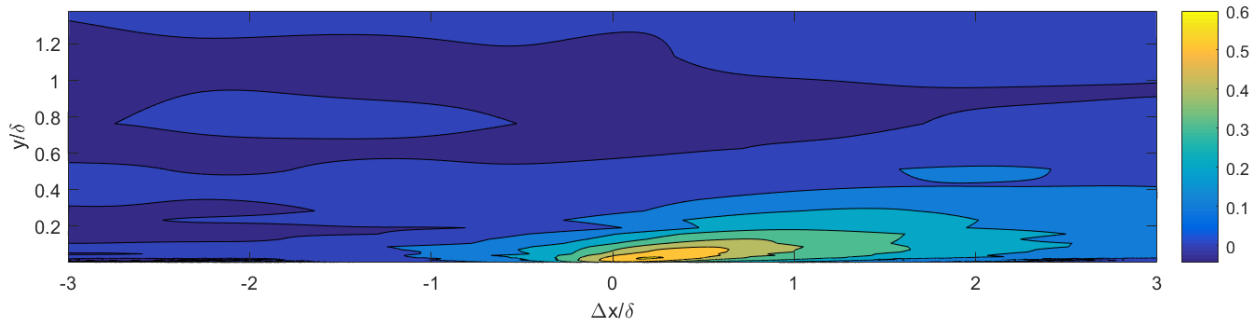

Figure 4: Cross correlation for all Y positions

The Figure shows large scale turbulent events, represented by the yellow high correlation regions. These large scale correlation events are inclined, as the hotwire sensor would have to be moved progressively downstream to match the peak of the correlation between the hotwire and hotfilm as the wall distance between these sensors is increased.

Additionally, as the hotwire sensor is moved away from the wall, the correlation decreases until there is effectively no correlation at about $0.5 \times \delta$. This suggests that the turbulent event either has a different structure further away from the wall, such that there is a low correlation between the hotwire and hotfilm sensors at larger distances; or the turbulent event only exists in the lower portion of the boundary layer.

Question 5

A conditional average of the $Y = 20$ hotwire signal was calculated based on a positive gradient of the hotfilm signal. Two adjacent points were used to calculate the gradient of the hotfilm signal. The code used, and a plot of the conditional average are shown below.

```
len = 1000; % Number of values to average either side

% Read and filter values
film = readBin('f',20);
wire = readBin('w',20);
film = lpFilter(film,T,N,50);

% Use two points for gradient
next_point = [film(2:end); 0];
pos_grad = (next_point - film) > 0;

% Average values if gradient of film is positive
condAverage = zeros(2*len+1,1);
nPoints = 0;
for i = len+1:N-len-1
    if (pos_grad(i))
        condAverage = condAverage + wire(i-len:i+len);
        nPoints = nPoints + 1;
    end
end
condAverage = condAverage/nPoints;

plot((-len:len)/Fs,condAverage);
xlabel('\Deltat (s)');
ylabel('Velocity (m/s)');
```

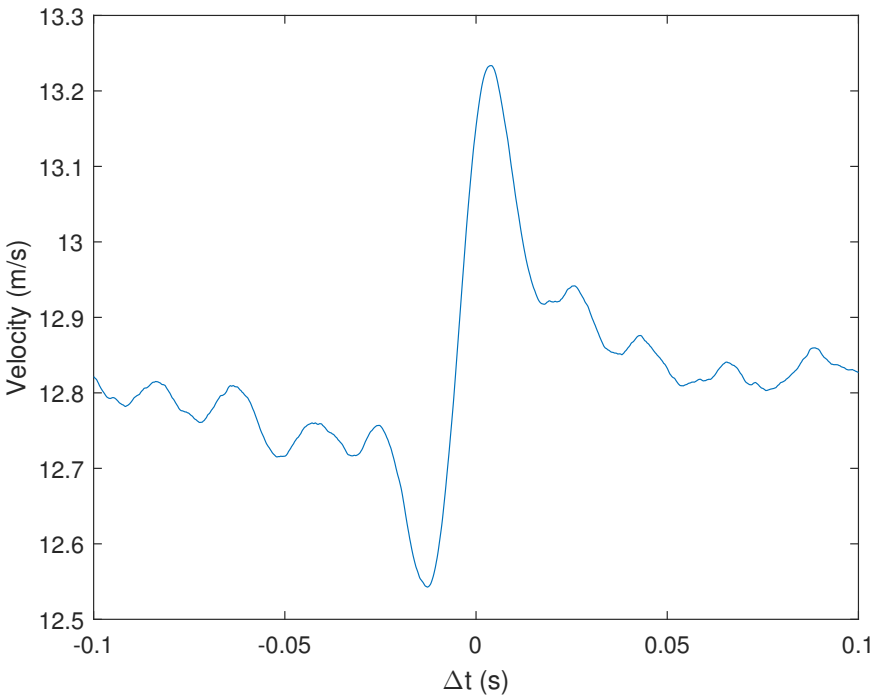

Figure 5: Conditional average of the hotwire signal

Question 6

The power spectral density was calculated using a single 30 second sample at $Y = 20$, without using intervals or overlapping.

```
% New constants
Fs    = 30*10^3; % Hz
T     = 30;      % s
N     = Fs*T;    % Number of samples
df    = Fs/N;

% Read file
wire = readBin('b',10);
wire = wire - mean(wire);

% Get Fourier transform
g = fft(wire)/N;
g = g(1:N/2+1);
Phi = 2.*g.*conj(g)/df;        % Phi = A^2/2 = 2*g*conj(g)

f = df*(0:N/2)';

% Plot spectrum
figure
semilogx(f,f.*Phi);
xlabel('f (Hz)');
ylabel('f\Phi');
```

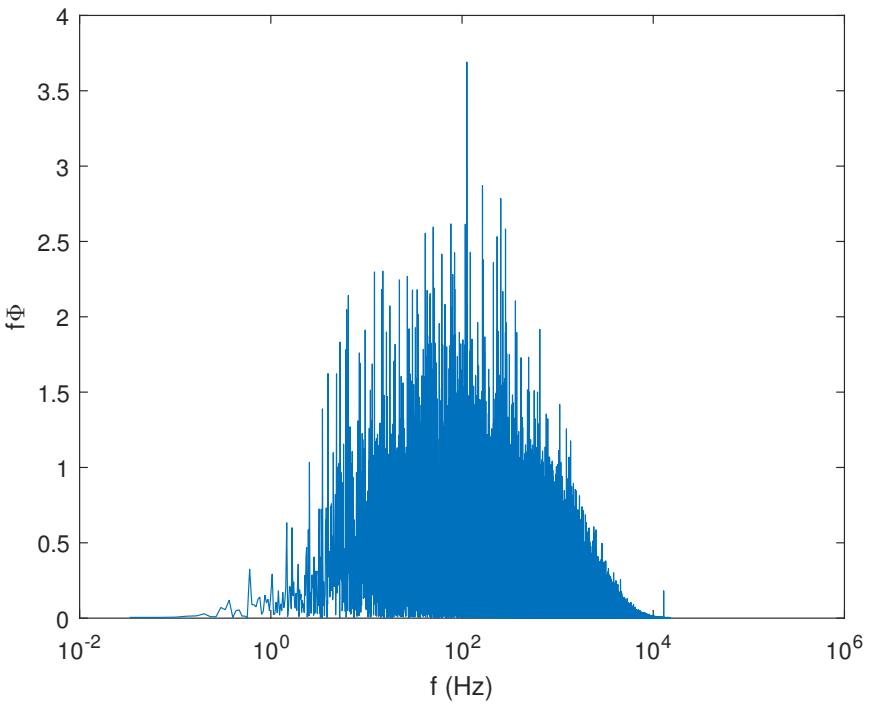

Figure 6: Hairy power spectral density

The spectrum above is still very hairy (has lots of large random spikes), which means that at the moment the spectrum is not well converged. However, the power spectral density approaches zero for very low frequencies, and also for very high frequencies, suggesting that all of the frequencies that carry power in the turbulent signal have been resolved. The sampling frequency is linked to the highest resolvable frequency, while the sampling time is linked to the lowest resolvable frequency. Since all of the frequencies of interest have been resolved, the sampling frequency as well as the sampling time are sufficient.

Note that there is high frequency spike just above 10000Hz. However this seems to be an outlier, as there are no other large spikes in the region.

Additionally, it can be shown that

$$\int_0^\infty \Phi(f)df = \frac{1}{N}\sum_{m=1}^{N} u_m^2 \tag{1.6.1}$$

This was checked using the following matlab code, where the trapezoidal numerical integration method is used

```
% Check if area under graph equals variance
Area_spectrum = trapz(f,Phi)
variance = mean(wire.^2)
```

Both the variables `Area_spectrum` and `variance` have the value 2.5695, confirming the identity. The values for `f`, `Phi`, and `wire` were used from the earlier code segment.

Question 7

To better converge the spectrum, the power spectral density was calculated for lots of subsets of the turbulent signal, and the results were averaged. A total of 10 sets of data were used, each with a duration of 30 seconds and sampled at 30kHz. For each of these data sets, 1000 sub-samples of 7 seconds were

taken and the spectrum was calculated. 7 seconds was chosen as this allowed the majority of the low frequency information to be retained, while still allowing for lots of independent data sets to be averaged. The code and spectrum are shown below.

```
T_total = 30;     % s
Fs      = 30*10^3; % Hz
T       = 7;       % s
N       = Fs*T;    % Number of samples
df      = Fs/N;
FILES = 10;

% Start index of each interval
n_intervals = 1000;
idx = 1:round(Fs*(T_total-T)/n_intervals):Fs*(T_total-T);

% Declare arrays
f   = df*(1:N/2)';
Phi = zeros(N/2,1);
nloops = FILES*n_intervals;
% Loop through all files
for n = 1:FILES
    % Read burst file
    u = readBin('b',n);

    % Loop through intervals with overlapping
    for i = 1:length(idx)

        % Extract section of signal
        ui = u(idx(i):N+idx(i)-1);
        ui = ui - mean(ui);

        % Get Fourier transform and add contribution
        g   = fft(ui)/N;
        Phi = Phi + 2.*g(2:(N/2+1)).*conj(g(2:(N/2+1)))./(df);
    end
end

% Average power spectral density
Phi = Phi/nloops;

% Plot spectrum
figure
semilogx(f,f.*Phi);
xlabel('f (Hz)');
ylabel('f\Phi');
```

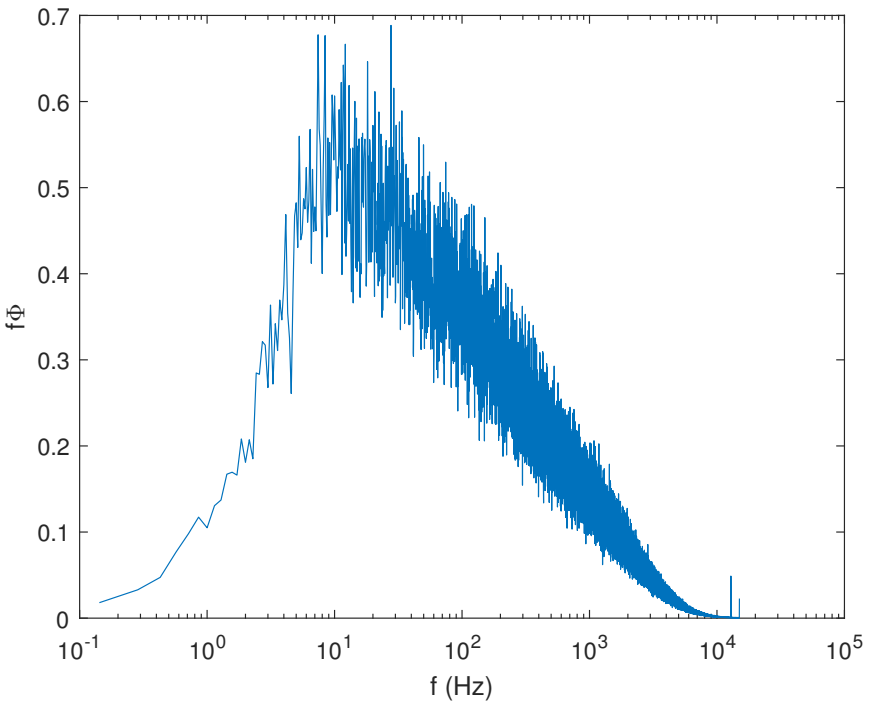

Figure 7: Converged power spectral density

Part 2 - Resistance of a developing smooth and rough-wall boundary layer

All of the MATLAB code used to complete this section is given in Appendix B

Question 1

The mean velocity profile for the smooth case is given as:

$$u^+ = \frac{u\left(y\right)}{U_\tau} = \frac{1}{\kappa}\log\left(\frac{yU_\tau}{\nu}\right) + A - \frac{1}{3\kappa}\left(\frac{y}{\delta}\right)^3 + \frac{\Pi}{\kappa}2\left(\frac{y}{\delta}\right)^2\left(3-2\left(\frac{y}{\delta}\right)\right) \tag{2.1.1}$$

Rewriting the equation for u^+ in terms of z^+ gives:

$$u^+ = \frac{1}{\kappa}\log\left(\frac{zU_\tau}{\nu}\right) + A - \frac{1}{3\kappa}\left(\frac{\frac{zU_\tau}{\nu}}{\frac{\delta U_\tau}{\nu}}\right)^3 + \frac{\Pi}{\kappa}2\left(\frac{\frac{zU_\tau}{\nu}}{\frac{\delta U_\tau}{\nu}}\right)^2\left(3-2\left(\frac{\frac{zU_\tau}{\nu}}{\frac{\delta U_\tau}{\nu}}\right)\right)$$

$$u^+ = \frac{1}{\kappa}\log\left(z^+\right) + A - \frac{1}{3\kappa}\left(\frac{z^+}{\delta^+}\right)^3 + \frac{\Pi}{\kappa}2\left(\frac{z^+}{\delta^+}\right)^2\left(3-2\left(\frac{z^+}{\delta^+}\right)\right) \tag{2.1.2}$$

u^+ could be calculated in MATLAB for each δ^+ and z^+. Values of z^+ were logarithmically spaced from near zero to δ^+ for each value of δ^+.

From u^+ the Reynolds number Re_θ based on the momentum thickness was calculated:

$$\theta = \frac{\nu}{U_\tau}\int_0^{\delta^+}\left(\frac{u^+}{S} - \frac{{u^+}^2}{S^2}\right)dz^+$$

$$\theta = \frac{\nu}{U_\infty}\int_0^{\delta^+}\left(u^+ - \frac{{u^+}^2}{S}\right)dz^+$$

$$Re_\theta = \frac{\theta U_\infty}{\nu} = \int_0^{\delta^+}\left(u^+ - \frac{{u^+}^2}{S}\right)dz^+ \tag{2.1.3}$$

Where $S = U_\infty/U_\tau = \frac{1}{\kappa}\log\left(\delta^+\right) + A - \frac{1}{3\kappa} + \frac{2\Pi}{\kappa}$. Note that $\Pi = 0.55$ for ZPG turbulent boundary layers.

A MATLAB script was written to numerically determine the momentum thickness (θ), as well as Re_θ for a range of values of δ^+. The script, as well as a plot of Re_θ vs $Re_\tau = \delta^+$ is shown below.

```
npoints = 1000;
A       = 5;
k       = 0.41;              % kappa
PI      = 0.55;              % For ZPG turbulent boundary layers
nu      = 8.97e-7;           % m^2/s
U_inf   = 30*0.51444444444; % m/s
ks      = 325e-6;            % m

%% Smooth Wall
```

```
% Get logarithmic spacing of d_plus
d_plus   = logspace(2,6,npoints);
Re_theta = zeros(size(d_plus));
Cf       = zeros(size(d_plus));

% Numerically evaluate momentum thickness at each d_plus
for i = 1:length(d_plus)
    % Create array to store incremental values of z_plus. Need to start
    % just above zero to keep value finite.
    z_plus = logspace(-4,log10(d_plus(i)),npoints);

    eta = z_plus./d_plus(i);

    % Calculate u_plus (smooth wall)
    u_plus = (1/k).*log(z_plus)+A-(1/(3*k)).*eta.^3 + ...
        (PI/k).*2.*(eta.^2).*(3-2.*eta);
    % Calculate S (smooth wall)
    S = (1/k).*log(d_plus(i))+A-(1/(3*k))+(2*PI/k);
    % Numerically integrate to get Re_theta
    dRe_theta = (u_plus - (u_plus.^2)./S);
    Re_theta(i) = trapz(z_plus,dRe_theta);
    % Calculate Cf
    Cf(i) = 2./(S.^2);
end

% Calculate theta
U_tau = U_inf.*sqrt(Cf./2);
theta = Re_theta.*nu./U_tau;
```

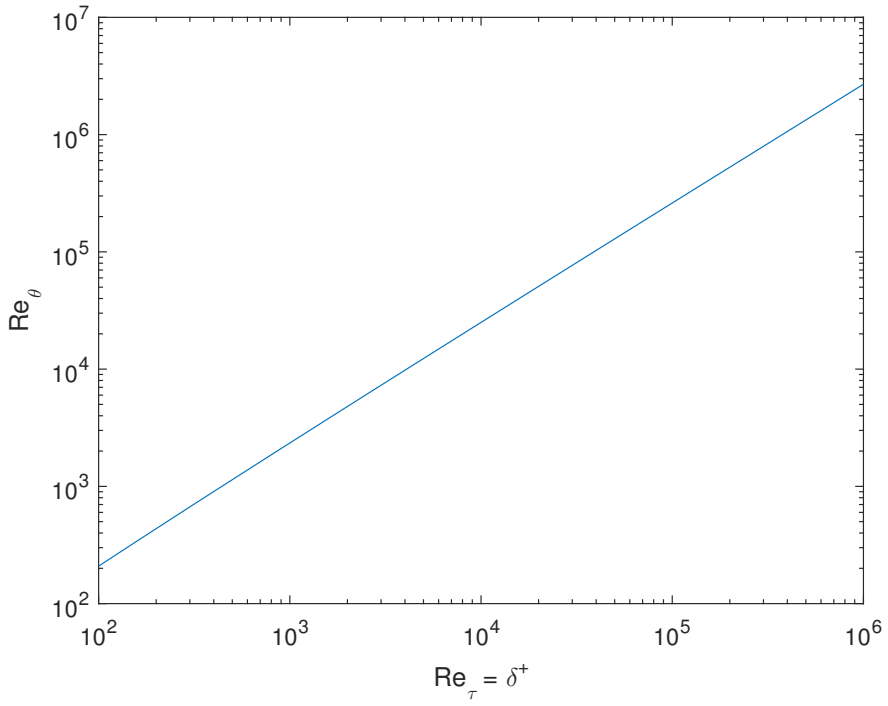

Figure 8: Re_θ vs Re_τ for a smooth wall

Question 2

The code from Question 1 also evaluates C_f for every value of δ^+ by using the following relationship:

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho U_\infty^2} = \frac{2}{S^2} \tag{2.2.1}$$

C_f vs $Re_\tau = \delta^+$ is plotted below.

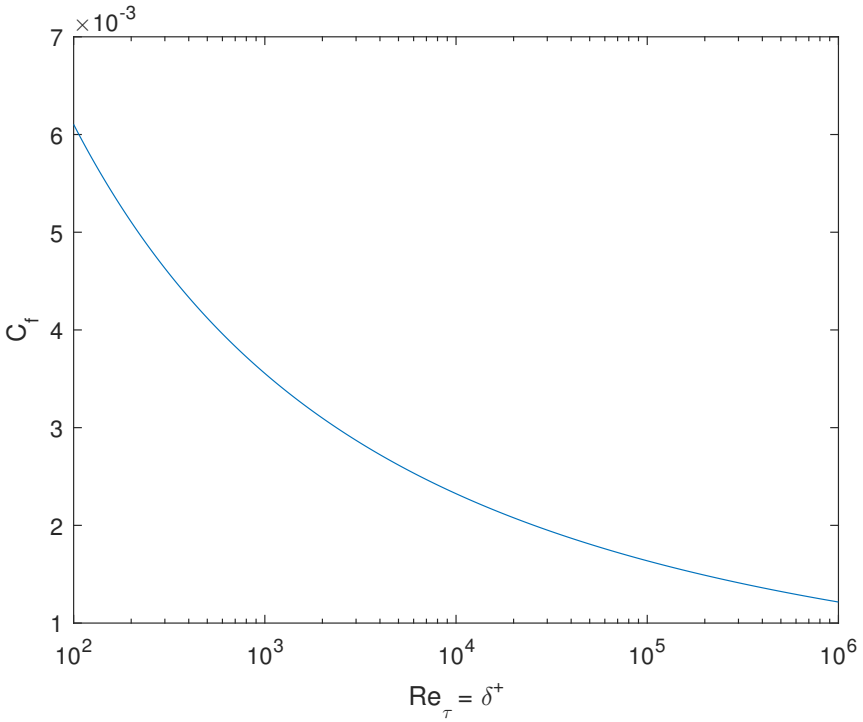

Figure 9: C_f vs Re_τ for a smooth wall

Question 3

By assuming a zero pressure gradient boundary layer, we can relate Re_θ to Re_x using the von Kármán momentum integral equation.

$$Re_x = \int \frac{2}{C_f} dRe_\theta \tag{2.3.1}$$

This integral is numerically evaluated in the code snippet below.

```
% Calculate Re_x
Re_x = cumtrapz(Re_theta, 2./Cf);
```

Although the integral does not start from zero (it starts from the value of Re_θ corresponding to the smallest value of δ^+), the missing part of the integral is very small and has a minimal effect on the overall value of Re_x. Now C_f vs Re_x can be plotted.

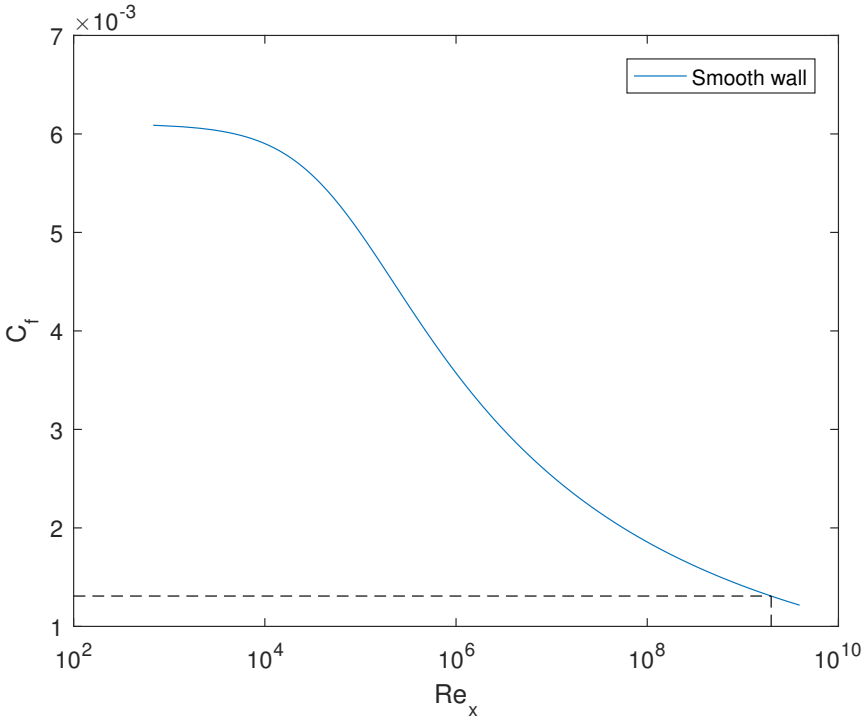

Figure 10: C_f vs Re_x for a smooth wall

At the back of a smooth submarine ($x = 115m$), $Re_x = 1.98 \times 10^9$. We can interpolate the values used in the figure above to find the corresponding friction coefficient at this location to be $C_f = 0.001308$. By rearranging the equation for S we can calculate δ^+, and then find the boundary layer thickness at this location to be $\delta = 1.2458m$. These values are summarised below:

At $x = 115m$ for a smooth submarine:

$$\begin{aligned} Re_x &= 1.9786 \times 10^9 \\ C_f &= 0.001308 \\ \delta &= 1.2458m \end{aligned}$$

The code used to determine these values is given below.

```
% Determine C_f and delta for x = 115 m
x = 115; %m
Re_x_115 = x*U_inf/nu;
Cf_115 = interp1q(Re_x', Cf', Re_x_115)
delta_plus_115 = exp(sqrt(2/Cf_115)*k - A*k + 1/3 - 2*PI);
delta_115 = delta_plus_115*nu*sqrt(2/Cf_115)/U_inf
```

Question 4

The mean velocity profile for the rough case is given as:

$$u^+ = \frac{u\left(y\right)}{U_\tau} = \frac{1}{\kappa}\log\left(\frac{yU_\tau}{\nu}\right) - \Delta U^+ + A - \frac{1}{3\kappa}\left(\frac{y}{\delta}\right)^3 + \frac{\Pi}{\kappa}2\left(\frac{y}{\delta}\right)^2\left(3 - 2\left(\frac{y}{\delta}\right)\right) \tag{2.4.1}$$

Where $\Delta U^+ = \frac{1}{\kappa}\log\left(k_s^+\right) + A - 8.5$

Rewriting the equation for $u^+$when $y = \delta$ gives:

$$\frac{U_\infty}{U_\tau} = \frac{1}{\kappa}\log\left(\delta^+\right) - \frac{1}{\kappa}\log\left(k_s^+\right) + 8.5 - \frac{1}{3\kappa} + \frac{2\Pi}{\kappa}$$
$$0 = \frac{1}{\kappa}\log\left(\delta^+\right) - \frac{1}{\kappa}\log\left(\frac{k_s U_\tau}{\nu}\right) + 8.5 - \frac{1}{3\kappa} + \frac{2\Pi}{\kappa} - \frac{U_\infty}{U_\tau} \tag{2.4.2}$$

For every δ^+, this equation was solved for U_τ in MATLAB using `fsolve`. Once U_τ was known, k_s^+ and u^+ could be calculated as shown below.

$$k_s^+ = \frac{k_s U_\tau}{\nu} \tag{2.4.3}$$
$$u^+ = \frac{1}{\kappa}\log\left(z^+\right) - \frac{1}{\kappa}\log\left(k_s^+\right) + 8.5 - \frac{1}{3\kappa}\left(\frac{z^+}{\delta^+}\right)^3 + \frac{\Pi}{\kappa}2\left(\frac{z^+}{\delta^+}\right)^2\left(3 - 2\left(\frac{z^+}{\delta^+}\right)\right) \tag{2.4.4}$$

From u^+ the Reynolds number Re_θ based on the momentum thickness was calculated as:

$$Re_\theta = \int_0^{\delta^+}\left(u^+ - \frac{u^+}{S}\right)dz^+ \tag{2.4.5}$$

Where $S = \frac{1}{\kappa}\log\left(\delta^+\right) - \frac{1}{\kappa}\log\left(k_s^+\right) + 8.5 - \frac{1}{3\kappa} + \frac{2\Pi}{\kappa}$.

The skin friction coefficient was determined as a function of Re_x for when the submarine (flat plate) has a roughness of $k_s = 325\mu m$. The code used to determine this is given in Appendix B. The smooth and rough curves are plotted below. Note that although the same range of δ^+ was used to generate each curve, this corresponds to a different range of Re_x for each case.

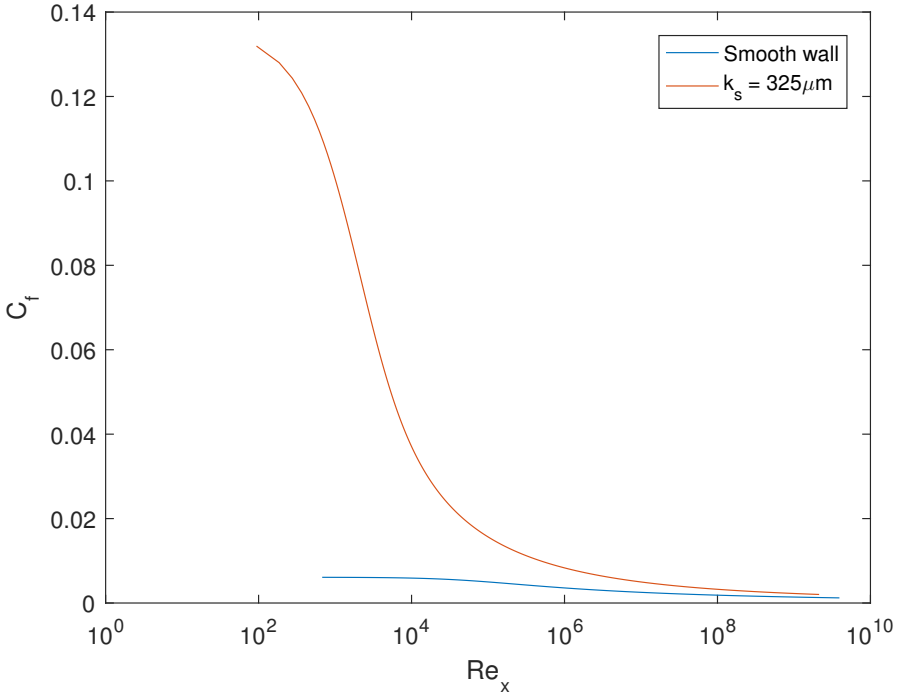

Figure 11: C_f vs Re_x

Zooming in we can compare the C_f values for both the smooth and the rough case at $x = 115m$.

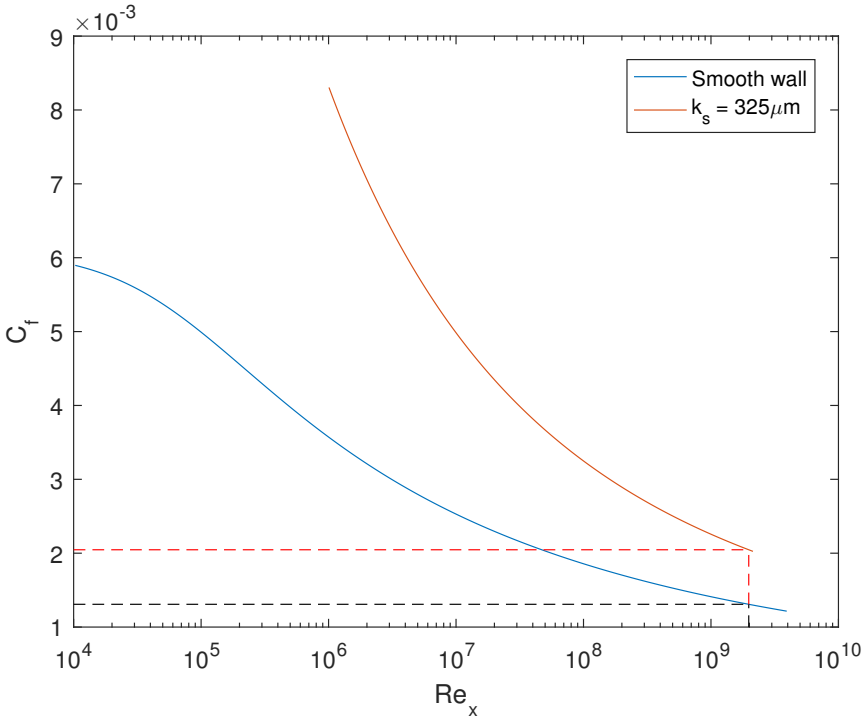

Figure 12: C_f vs Re_x

The skin friction coefficient and boundary layer thickness values are compared in the following table. Note that the roughness significantly increases the boundary layer thickness, and increases the skin friction coefficient at $x = 115m$ by approximately 1.5 times.

	Smooth Wall	Rough Wall
C_f	0.001308	0.002046
δ	$1.2458m$	$1.7062m$

Table 1: Comparison of values

A Part 1 Code

A.1 readBin.m

```
% MCEN90018:   Advanced Fluid Dynamics – Assignment 3
% ------------------------------------------------------------------------
% Mischka      Kamener    539030                    Last modified: 28/5/16
% Robert       Haberkern  637517
%
% Custom function to read the .bin files

function A = readBin(type, num)

if type == 'w' || type == 'f'
    fp = fopen(sprintf('Data/u_h%s_ypos%d.bin', type, num));
else
    fp = fopen(sprintf('Data/u_hw_ypos20_burst%d.bin', num));
end
A = fread(fp, 'float');
fclose(fp);
```

A.2 lpFilter.m

```
% MCEN90018:   Advanced Fluid Dynamics – Assignment 3
% ------------------------------------------------------------------------
% Mischka      Kamener    539030                    Last modified: 28/5/16
% Robert       Haberkern  637517
%
% Low pass filter that takes a signal of N samples with a total length of T
% seconds, and the cutoff frequency in Hz.

function filt = lpFilter(signal,T,N,cutoff)

% Get Fourier transform, and find cutoff index
g = fft(signal);
deltaf = 1/T;
f = ([0:1:N/2 (N/2-1):-1:1])'*deltaf; % Frequencies of each coefficient

% Set frequencies lower than cutoff to zero, and reverse transform
g(f>cutoff) = 0;
filt = real(ifft(g));
```

A.3 Part1.m

```
% MCEN90018:   Advanced Fluid Dynamics – Assignment 3
% ------------------------------------------------------------------------
% Mischka      Kamener    539030                    Last modified: 28/5/16
% Robert       Haberkern  637517
%
% Script for Part 1

%% Constants
delta = 0.326;    % m
Fs    = 10*10^3; % Hz
T     = 30;       % s
N     = Fs*T;     % Number of samples
```

```
y_i   = 0.0002;  % m
y_f   = 0.450;   % m
N_POS = 40;      % Number of data points

%% Question 1

% Read file
f20 = readBin('f',20);

f20_filtered = lpFilter(f20,T,N,50);

% Plot original and filtered signal over 2 seconds
idx = 10000:20000;
figure
hold on
plot(idx/Fs,f20(idx),'LineWidth',1);
plot(idx/Fs,f20_filtered(idx),'LineWidth',2);
hold off
xlabel('t (s)');
ylabel('Velocity (m/s)');
legend('Unfiltered', 'Filtered');
title('Hotfilm signal for Y=20');

%% Question 2

% Read file
f20 = readBin('f',20);
w20 = readBin('w',20);

f20 = lpFilter(f20,T,N,50);

% Find mean
Uc  = mean(w20);

% Adjust for mean and standard deviation
f20 = f20 - mean(f20); f20 = f20./std(f20);
w20 = w20 - mean(w20); w20 = w20./std(w20);

% Zero pad vector to be shifted
w20_p = [zeros(size(w20)); w20; zeros(size(w20))];
R_cc  = zeros(size(-N:N-1))';

tic
% Compute cross correlation. Slide one signal across the other
for n = -N:N-1;
    R_cc(n+N+1) = sum(f20.*w20_p((N+1:2*N)+n));
end
toc

% Normalise by number of elements
R_cc = R_cc./N;

% Determine dt values to plot
lim = 3*delta/Uc;
dt = (-N:N-1)./Fs;
dx = -dt*Uc;

% Plot graph
plot(dx(abs(dt)<lim)/delta,R_cc(abs(dt)<lim))
xlabel('\Deltax/\delta');
ylabel('R');

%% Question 3

% Read files
f20 = readBin('f',20);
```

```
w20 = readBin('w',20);

f20 = lpFilter(f20,T,N,50);

% Find mean
Uc  = mean(w20);

% Adjust for mean and standard deviation
f20 = f20 - mean(f20); f20 = f20./std(f20);
w20 = w20 - mean(w20); w20 = w20./std(w20);

% Pad with zeros
f20_p = [zeros(size(f20)); f20];
w20_p = [zeros(size(w20)); w20];

tic
% Compute cross correlation using fft (R_fft goes from -N to N-1)
R_fft = fftshift(ifft(conj(fft(f20_p)).*(fft(w20_p))));
toc

% Normalise by number of elements
R_fft = R_fft./N;

% Determine dt values to plot
lim = 3*delta/Uc;
dt = (-N:N-1)./Fs;
dx = -dt*Uc;

% Plot graph
plot(dx(abs(dt)<lim)/delta,R_fft(abs(dt)<lim),'-', ...
    dx(abs(dt)<lim)/delta,R_cc(abs(dt)<lim),'--')
xlabel('\Deltax/\delta');
ylabel('R');
legend('FFT method', 'Long-hand method');

%% Question 4

% Create meshgrid and initialise matrices
[dt, y] = meshgrid((-N:N-1)./Fs, logspace(log10(y_i), log10(y_f), N_POS));
R = zeros(size(dt));
Uc = zeros(size(dt));

% Loop through all files
for i = 1:N_POS
    % Read file
    film = readBin('f',i);
    wire = readBin('w',i);
    % Find mean
    Uc(i,:) = mean(wire);

    film = lpFilter(film,T,N,50);

    % Adjust for mean and standard deviation
    film = film - mean(film); film = film./std(film);
    wire = wire - mean(wire); wire = wire./std(wire);

    % Pad with zeros
    film_p = [zeros(size(film)); film];
    wire_p = [zeros(size(wire)); wire];

    % Compute cross correlation using fft (R_fft goes from -N to N-1)
    R(i,:) = fftshift(ifft(conj(fft(film_p)).*(fft(wire_p)))./N;
end

% Determine dt values to plot
lim = 3*delta./Uc;
```

```
dx = -dt.*Uc;
inPlot = abs(dt)<lim;

% Create new grid so that contours can be drawn
[xx, yy] = meshgrid(-3:0.001:3, logspace(log10(y_i), log10(y_f), N_POS));
zz = griddata(dx(inPlot)/delta,y(inPlot),R(inPlot),xx,yy);

figure
contourf(xx,yy/delta,zz);
xlabel('\Deltax/\delta');
ylabel('y/\delta');
axis equal
colorbar

%% Question 5

len = 1000; % Number of values to average either side

% Read and filter values
film = readBin('f',20);
wire = readBin('w',20);
film = lpFilter(film,T,N,50);

% Use two points for gradient
next_point = [film(2:end); 0];
pos_grad = (next_point - film) > 0;

% Average values if gradient of film is positive
condAverage = zeros(2*len+1,1);
nPoints = 0;
for i = len+1:N-len-1
    if (pos_grad(i))
        condAverage = condAverage + wire(i-len:i+len);
        nPoints = nPoints + 1;
    end
end
condAverage = condAverage/nPoints;

plot((-len:len)/Fs,condAverage);
xlabel('\Deltat (s)');
ylabel('Velocity (m/s)');

%% Question 6

% New constants
Fs    = 30*10^3; % Hz
T     = 30;      % s
N     = Fs*T;    % Number of samples
df    = Fs/N;

% Read file
wire = readBin('b',10);
wire = wire - mean(wire);

% Get Fourier transform
g = fft(wire)/N;
g = g(1:N/2+1);
Phi = 2.*g.*conj(g)/df;        % Phi = A^2/2 = 2*g*conj(g)

f = df*(0:N/2)';

% Plot spectrum
figure
semilogx(f,f.*Phi);
xlabel('f (Hz)');
ylabel('f\Phi');
```

```
% Check if area under graph equals variance
Area_spectrum = trapz(f,Phi)
variance = mean(wire.^2)

%% Question 7

% New constants
T_total = 30;    % s
Fs    = 30*10^3; % Hz
T     = 7;       % s
N     = Fs*T;    % Number of samples
df    = Fs/N;
FILES = 10;

% Start index of each interval
n_intervals = 1000;
idx = 1:round(Fs*(T_total-T)/n_intervals):Fs*(T_total-T);

% Declare arrays
f   = df*(1:N/2)';
Phi = zeros(N/2,1);
nloops = FILES*n_intervals;
% Loop through all files
for n = 1:FILES
    % Read burst file
    u = readBin('b',n);

    % Loop through intervals with overlapping
    for i = 1:length(idx)

        % Extract section of signal
        ui = u(idx(i):N+idx(i)-1);
        ui = ui - mean(ui);

        % Get Fourier transform and add contribution
        g   = fft(ui)/N;
        Phi = Phi + 2.*g(2:(N/2+1)).*conj(g(2:(N/2+1)))./(df);
    end
end

% Average power spectral density
Phi = Phi/nloops;

% Plot spectrum
figure
semilogx(f,f.*Phi);
xlabel('f (Hz)');
ylabel('f\Phi');
```

B Part 2 Code

B.1 Part2.m

```
% MCEN90018:   Advanced Fluid Dynamics - Assignment 3
% ------------------------------------------------------------------------
% Mischka      Kamener    539030                    Last modified: 28/5/16
% Robert       Haberkern  637517
%
% Script for Part 2
```

```
npoints = 1000;
A       = 5;
k       = 0.41;             % kappa
PI      = 0.55;             % For ZPG turbulent boundary layers
nu      = 8.97e-7;          % m^2/s
U_inf   = 30*0.51444444444; % m/s
ks      = 325e-6;           % m

%% Smooth Wall
% Get logarithmic spacing of d_plus
d_plus   = logspace(2,6,npoints);
Re_theta = zeros(size(d_plus));
Cf       = zeros(size(d_plus));

% Numerically evaluate momentum thickness at each d_plus
for i = 1:length(d_plus)
    % Create array to store incremental values of z_plus. Need to start
    % just above zero to keep value finite.
    z_plus = logspace(-4,log10(d_plus(i)),npoints);

    eta = z_plus./d_plus(i);

    % Calculate u_plus (smooth wall)
    u_plus = (1/k).*log(z_plus)+A-(1/(3*k)).*eta.^3 + ...
        (PI/k).*2.*(eta.^2).*(3-2.*eta);
    % Calculate S (smooth wall)
    S = (1/k).*log(d_plus(i))+A-(1/(3*k))+(2*PI/k);
    % Numerically integrate to get Re_theta
    dRe_theta = (u_plus - (u_plus.^2)./S);
    Re_theta(i) = trapz(z_plus,dRe_theta);
    % Calculate Cf
    Cf(i) = 2./(S.^2);
end

% Calculate theta
U_tau = U_inf.*sqrt(Cf./2);
theta = Re_theta.*nu./U_tau;

% Calculate Re_x
Re_x = cumtrapz(Re_theta, 2./Cf);

% Determine C_f and delta for x = 115 m
x = 115; %m
Re_x_115 = x*U_inf/nu;
Cf_115 = interp1q(Re_x', Cf', Re_x_115)
delta_plus_115 = exp(sqrt(2/Cf_115)*k - A*k + 1/3 - 2*PI);
delta_115 = delta_plus_115*nu*sqrt(2/Cf_115)/U_inf

% Plot figures
figure
loglog(d_plus,Re_theta)
ylabel('Re_{\theta}');
xlabel('Re_{\tau} = \delta^+');

figure
semilogx(d_plus, Cf)
ylabel('C_f');
xlabel('Re_{\tau} = \delta^+');

figure
semilogx(Re_x,Cf)
hold on
semilogx([100 Re_x_115], [Cf_115 Cf_115], '--k')
semilogx([Re_x_115 Re_x_115], [0.001 Cf_115], '--k')
hold off
ylabel('C_f');
```

```
xlabel('Re_{x}');
legend('Smooth wall');

%% Rough Wall
% Get logarithmic spacing of d_plus
Re_theta_r = zeros(size(d_plus));
Cf_r       = zeros(size(d_plus));
U_tau_r    = zeros(size(d_plus));

% Turn off display for fsolve
options = optimset('Display', 'off');

% Calculate Cf for rough wall
for i = 1:length(d_plus)
    % Create function to solve for Ut
    U_tauFun = @(Ut) (1/k)*log(d_plus(i)) - (1/(3*k))+(2*PI/k) ...
        - (1/k)*log(ks*Ut/nu) + 8.5 - U_inf/Ut;
    % Get initial guss for Ut
    guess = 0.1;
    if i ~= 1
        guess = U_tau_r(i-1);
    end
    % Calculate U_tau
    U_tau_r(i) = fsolve(U_tauFun, guess, options);

    % Create array to store incremental values of z_plus. Need to start
    % just above zero to keep value finite.
    z_plus = logspace(-4,log10(d_plus(i)),npoints);

    eta = z_plus./d_plus(i);

    % Calculate ks_plus
    ks_plus = ks*U_tau_r(i)/nu;
    % Calculate u_plus (rough wall)
    u_plus = (1/k).*log(z_plus)-(1/k).*log(ks_plus)+8.5 ...
        -(1/(3*k)).*eta.^3 + (PI/k).*2.*(eta.^2).*(3-2.*eta);
    % Calculate S (rough wall)
    S = (1/k).*log(d_plus(i))-(1/k).*log(ks_plus)+8.5 ...
        -(1/(3*k))+(2*PI/k);
    % Numerically integrate to get Re_theta
    dRe_theta = (u_plus - (u_plus.^2)./S);
    Re_theta_r(i) = trapz(z_plus,dRe_theta);
    % Calculate Cf
    Cf_r(i) = 2./(S.^2);

end

% Calculate Re_x
Re_x_r = cumtrapz(Re_theta_r, 2./Cf_r);

% Determine C_f and delta for x = 115 m (rough wall)
Cf_115_r = interp1q(Re_x_r', Cf_r', Re_x_115)
delta_plus_115_r = exp(sqrt(2/Cf_115_r)*k + ...
    log(ks*U_inf*sqrt(Cf_115_r/2)/nu) - 8.5*k + 1/3 - 2*PI);
delta_115_r = delta_plus_115_r*nu*sqrt(2/Cf_115_r)/U_inf

% Plot results
figure
semilogx(Re_x, Cf, Re_x_r, Cf_r);
ylabel('C_f');
xlabel('Re_{x}');
legend('Smooth wall', 'k_s = 325\mum');

inPlot = Re_x > 1e4;
inPlot_r = Re_x_r > 1e6;
```

```
figure
semilogx(Re_x(inPlot), Cf(inPlot), Re_x_r(inPlot_r), Cf_r(inPlot_r));
hold on
semilogx([10000 Re_x_115], [Cf_115 Cf_115], '--k')
semilogx([Re_x_115 Re_x_115], [0.001 Cf_115], '--k')
semilogx([10000 Re_x_115], [Cf_115_r Cf_115_r], '--r')
semilogx([Re_x_115 Re_x_115], [Cf_115 Cf_115_r], '--r')
hold off
ylabel('C_f');
xlabel('Re_{x}');
legend('Smooth wall', 'k_s = 325\mum');
```