```
clear all
close all
clc
```

Designing the TUD Multiphase-flow Tunnel

Geometry specifications

Firstly, geometry parameters require to be loaded. An array containing the static pressures and the cross sectional areas are necessary. Also, it must be recognised where the intake and outlet of the Pipe section exist, in order to account it in the flow-rate calculation.

In this section, we also calculate the volumetric capacity of the tunnel for 2 reasons: We get an idea of how heavy the tunnel will be and how much heat would be required to raise its temperature by 1 degree celcius. The latter being useful in judging whether a heat-exchanger would be necessary.

```
% Import main tunnel parameters
NN = 10;
TunnelParams = importtunnelparams('Tunnel_params.csv', 1, 13)
```

```
TunnelParams =
    'Straight'    'Straight'    'Bend'                       'Straight'    'Straight'    'Adapter'     'E
    'pipe'        'pipe'        'pipe'                       'pipe'        'pipe'        'C2R'         'c
    '2'           '7'           '0'                          '4'           '1'           '0.2179'      '0
    '0.6'         '1.5'         '1.5'                        '1.5'         '1.5'         '0.8282'      '0
    '1.5'         '1.5'         '0'                          '1.5'         '0.8282'      '0.7348'      '0
    '0'           '0'           '0'                          '0'           '0'           '0'           '0
    '0'           '0'           '0'                          '0'           '0'           '0.7348'      '0
    '7'           '7'           '7'                          '6.25'        '2.25'        '1.25'        '1
    '7'           '7'           '6.25'                       '2.25'        '1.25'        '1.0321'      '0
    '0'           '0'           '0'                          '0'           '0'           '0'           '0
    '0'           '0'           '0'                          '0'           '0'           '0'           '0
    '0'           '0'           'profiled_rounded_normal'    '0'           '0'           '0'           't
    '0'           '0'           '0'                          '0'           '0'           '0'           '0
```

```
% Use the geom function to output a component matrix for both the main tunnel and the Pipe sec
[ComponentsMain,~,~,TunnelVolume, F0byFiMain, MainArInlet] = geom_tunnel(TunnelParams,NN);
```

```
Straight pipe
Straight pipe
Pipe bend
Straight pipe
Straight pipe
Circle to Square adapter
Ducted bend
Straight duct
Mesh in a duct
Dynamometer channel
Mesh in a duct
Ducted contraction
Straight duct
Straight duct
Straight duct
Ducted bend
Circle to Square adapter
```

```
Straight pipe
Pipe bend
Straight pipe
Straight pipe
```

```
% Import pipe section parameters
PipeParams = importpipeparams('Pipe_params.csv',1,13)
```

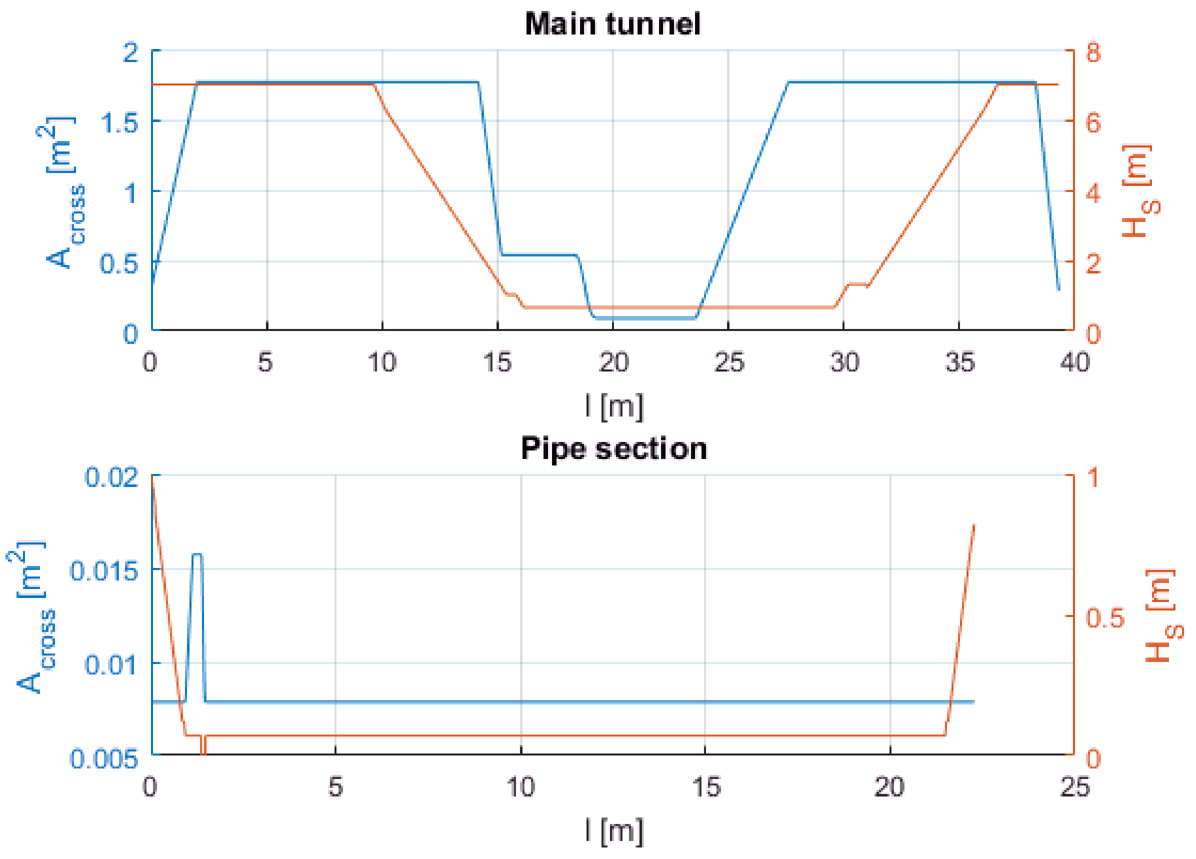
```
PipeParams =
    'Intake'    'Bend'         'Straight'    'Bend'                        'Straight'    'Mesh'       'Hor
    '3'         'arbitrary'    'pipe'        'pipe'                        'pipe'        'pipe'       'pip
    '0.1'       '0'            '0.7293'      '0'                           '0.2'         '0.075'      '0.0
    '0.1'       '0.1'          '0.1'         '0.1'                         '0.1'         '0.1414'     '0.1
    '0'         '160'          '0.1'         '0'                           '0.1414'      '0'          '0'
    '0'         '0'            '0'           '0'                           '0'           '0'          '0'
    '0'         '0'            '0'           '0'                           '0'           '0'          '0'
    '1'         '0.9'          '0.85'        '0.1207'                      '0.0707'      '0.0707'     '0.0
    '0.9'       '0.85'         '0.1207'      '0.0707'                      '0.0707'      '0.0707'     '0.0
    '30'        '0.1'          '0'           '0'                           '0'           '0.0007'     '0.1
    '0'         '0'            '0'           '0'                           '0'           '0.002'      '0.0
    '0'         '0'            '0'           'profiled_rounded_normal'     '0'           '0'          '0.0
    '0'         '0'            '0'           '0'                           '0'           '0'          '0.0
```

```
% Use the geom function to output a component matrix for both the main tunnel and the Pipe sec
[ComponentsPipe,intake,remerge,PipeVolume, F0byFiPipe, PipeArInlet] = geom_tunnel(PipeParams,N
```

```
Intake pipe for pipe section
Arbitrary bend
Straight pipe
Pipe bend
Straight pipe
Mesh in a pipe
Mesh in a pipe
Conical contraction
Straight pipe
Pipe bend
Straight pipe
Merging connection for pipe
```

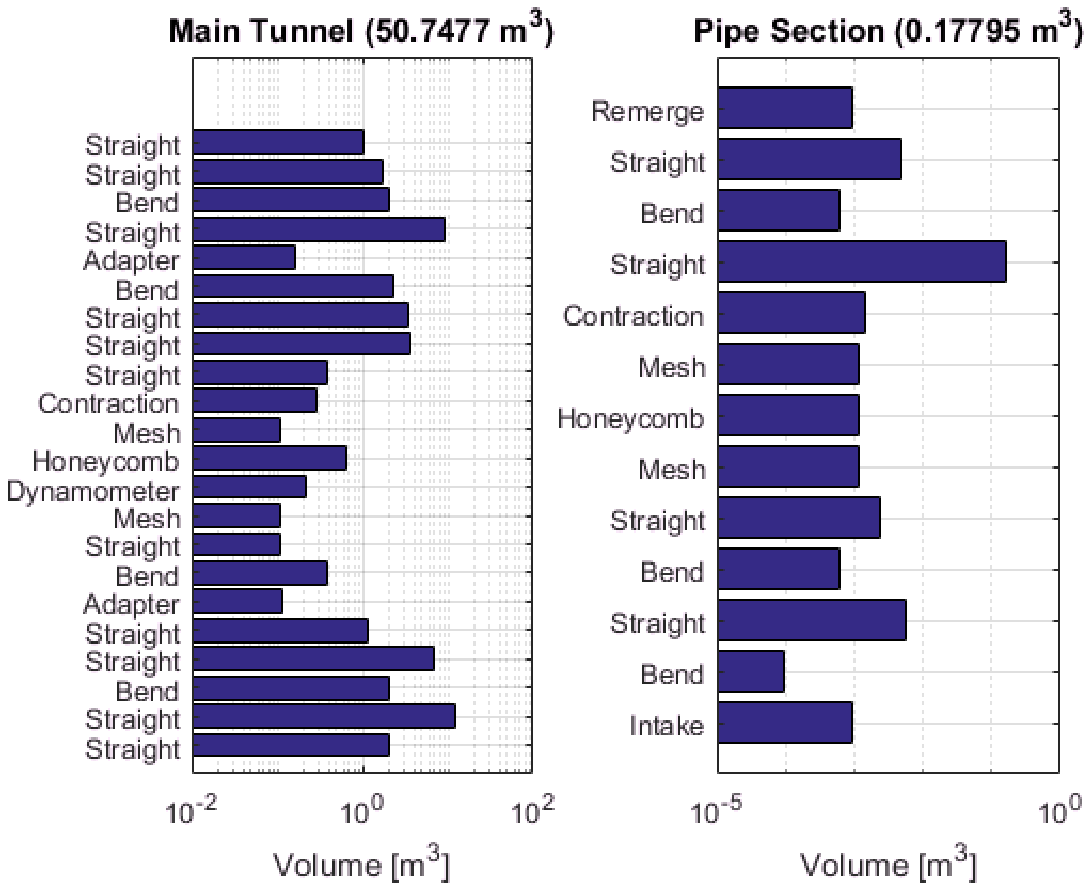
```
% Plot geometry parameters
figure(1)
subplot(2,1,1)
title('Main tunnel')
xlabel('l [m]')
hold on

yyaxis left
ylabel('A_{cross} [m^2]')
plot(ComponentsMain(1,:),ComponentsMain(2,:))

yyaxis right
ylabel('H_S [m]')
plot(ComponentsMain(1,:),ComponentsMain(5,:))

hold off
grid on

subplot(2,1,2)
title('Pipe section')
xlabel('l [m]')
```

```
hold on

yyaxis left
ylabel('A_{cross} [m^2]')
plot(ComponentsPipe(1,:),ComponentsPipe(2,:))

yyaxis right
ylabel('H_S [m]')
plot(ComponentsPipe(1,:),ComponentsPipe(5,:))

hold off
grid on
```



```
% Plotting tunnel volumes
figure(2)
subplot(1,2,1)
barh(TunnelVolume(:,1))
title(['Main Tunnel (' num2str(sum(TunnelVolume(:,1))) ' m^3)'])
xlabel('Volume [m^3]')
% ylim([min(TunnelVolume) max(TunnelVolume)])
set(gca, 'YTick', 1:22, 'YTickLabel', TunnelParams(1,:));
grid on
set(gca,'xscale','log')
% xtickangle(90)
subplot(1,2,2)
barh(PipeVolume(:,1))
set(gca, 'YTick', 1:13, 'YTickLabel', PipeParams(1,:));
set(gca,'xscale','log')
%xtickangle(90)
title(['Pipe Section (' num2str(sum(PipeVolume(:,1))) ' m^3)'])
% ylim([min(PipeVolume) max(PipeVolume)])
```

```
xlabel('Volume [m^3]')
grid on
```



Applying Continuity to calculate flow-rate in the Main tunnel and the Pipe section

Remember: Since we are designing a closed loop system, the pump only has to do work to overcome frictional losses in the circuit. Therefore, for a particular discharge, the operating head of the pump must correspond to the frictional losses in that flow-rate.

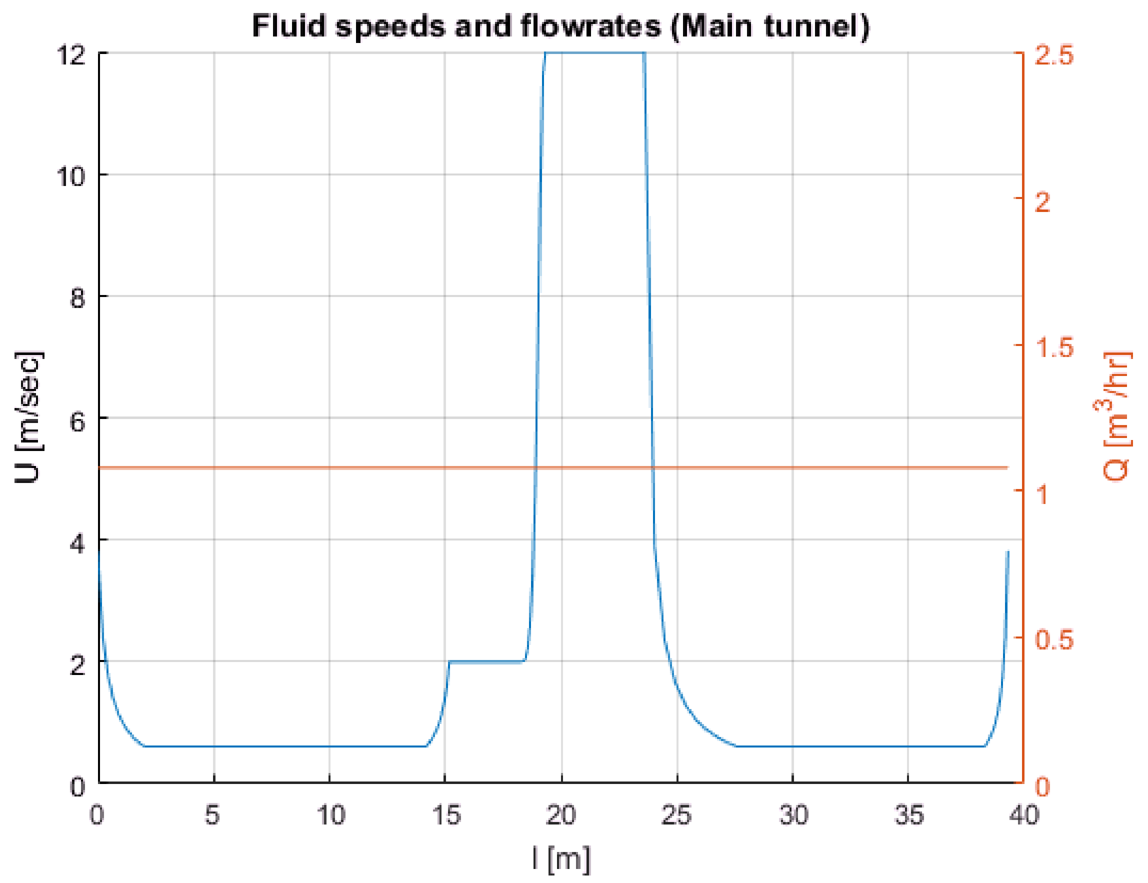
We want to control the flow within the pipe-section by simply introducing a solidity in the main tunnel, thereby running both sections simultaneously. Noting the point of intake of the pipe-section and the point where the pipe-section rejoins the main tunnel, the flow-rate in the tunnel can be estimated for a particular value of solidity in the main tunnel test-section. Therefore, the tunnel is controlled by 2 factors, the solidity at the test-section and the flow rate. When solidity is zero, we assume that the pipe-section is closed off. When the solidity is non-zero, but less than 1, we introduce flow into the pipe-section by continuity.

More importantly, when the solidity is non-zero, a new friction factor caused by the points of intake and re-combining of the pipe-section must be accounted.

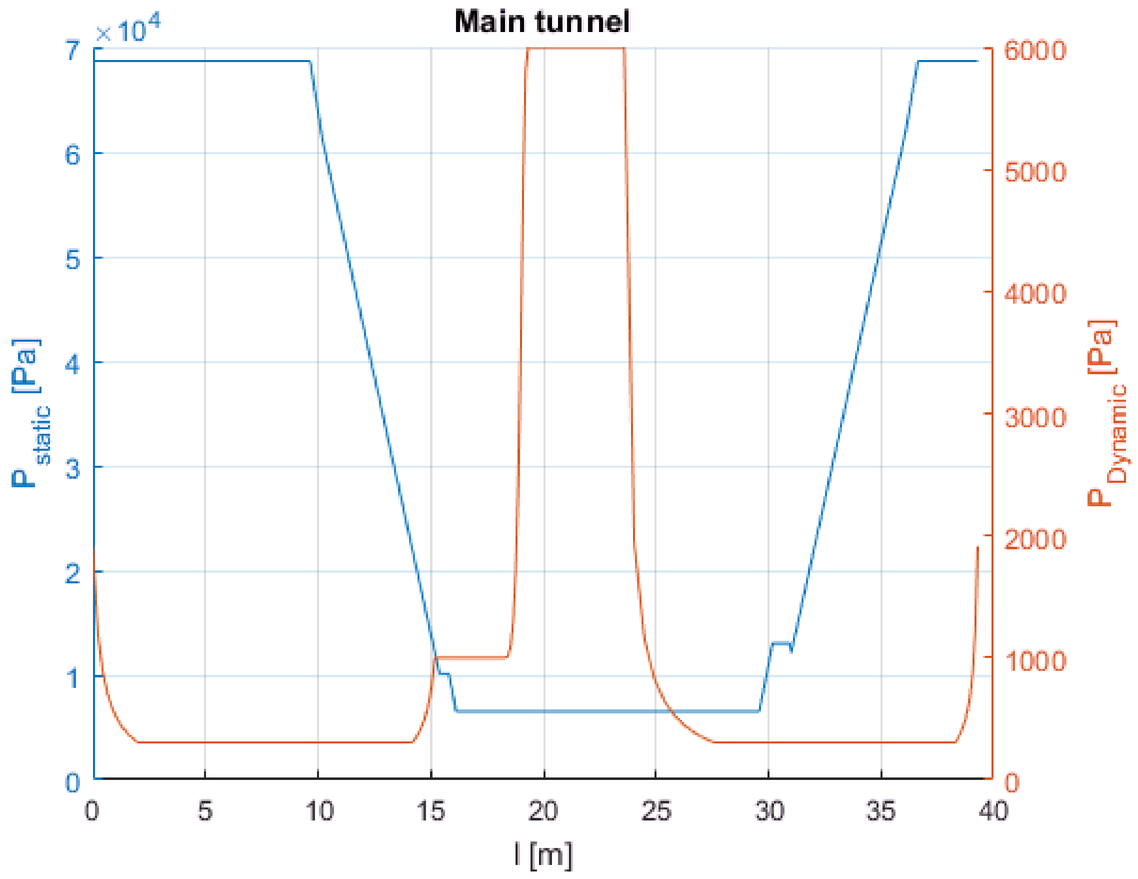
Also, when we use the pipe-section, since the pipe section is above the main tunnel, a head difference between the main test-section centreline and the pipe section centreline must be added.

```
% Calculating pressures from Bernoulli for given flow-rate and main test-section solidity

% Input flow parameters
ValveDia     = 0.05;                                      % Valve height
theta        = deg2rad(0);                               % Valve opening angle
solidity     = sin(theta) * ValveDia / 0.3;               % Test-section solidity
```

```
QPump        = 1.08;                                    % Pump output flowrate [m^2/hr]
rho          = 1000;                                    % Fluid density [kg/m^3]
SetP         = 10e5;                                    % Tunnel set pressure [Pascals]
nu           = 1e-6;                                    % Fluid kinematic viscosity [m^2/sec]


% Dividing the flow-rates as per solidity
QTunnel      = zeros(size(ComponentsMain(2,:)));
QPipe        = ones(size(ComponentsPipe(2,:))) * (QPump * solidity);
for i = 1:length(QTunnel)
   if ComponentsMain(1,i) < intake
       QTunnel(i) = QPump;
   elseif (ComponentsMain(1,i) >= intake && ComponentsMain(1,i) <= remerge)
       QTunnel(i) = QPump*(1-solidity);
   elseif ComponentsMain(1,i) > intake
       QTunnel(i) = QPump;
   end
end

%Plotting flow-rates
if solidity > 0
    figure(3)
    subplot(2,1,1)
    hold on
    plot(ComponentsMain(1,:),(QTunnel./ComponentsMain(2,:)))
    title('Fluid speeds and flowrates (Main tunnel)')
    xlabel('l [m]')
    ylabel('U [m/sec]')
    yyaxis right
    ylabel('Q [m^3/hr]')
    plot(ComponentsMain(1,:),QTunnel)
    grid on


    subplot(2,1,2)
    hold on
    plot(ComponentsPipe(1,:),(QPipe./ComponentsPipe(2,:)))
    title('Pipe section')
    xlabel('l [m]')
    ylabel('U [m/sec]')
    yyaxis right
    ylabel('Q [m^3/hr]')
    plot(ComponentsPipe(1,:),QPipe)
    grid on
    hold off
elseif solidity ==0
    figure(3)
    hold on
    plot(ComponentsMain(1,:),(QTunnel./ComponentsMain(2,:)))
    title('Fluid speeds and flowrates (Main tunnel)')
    xlabel('l [m]')
    ylabel('U [m/sec]')
    yyaxis right
    ylabel('Q [m^3/hr]')
    plot(ComponentsMain(1,:),QTunnel)
    grid on
end
```



```matlab
% Adjusting the water-head when solidity is non-zero
if solidity > 0
   HeadDifference      = min(ComponentsPipe(5,:)) - min(ComponentsMain(5,:));
   ComponentsPipe(5,:) = ComponentsPipe(5,:) + HeadDifference;
end

% Calculating the Static pressure [Pa] for a particular operating fluid density
PStatMain = ComponentsMain(5,:) .* rho * 9.81;

% Calculating the Dynamic pressure [Pa] of the flow in both sections
PDynMain  = (0.5*rho).* QTunnel./(ComponentsMain(2,:));

% Plotting the Static and Dynamic pressures
if solidity > 0
    PDynPipe  = (0.5*rho).* QPipe./(ComponentsPipe(2,:));
    PStatPipe = ComponentsPipe(5,:) .* rho * 9.81;

    figure(4)
    subplot(2,1,1)
    title('Main tunnel')
    xlabel('l [m]')
    hold on

    yyaxis left
    ylabel('P_{static} [Pa]')
    plot(ComponentsMain(1,:),PStatMain)

    yyaxis right
    ylabel('P_{Dynamic} [Pa]')
    plot(ComponentsMain(1,:),PDynMain)
    grid on
    hold off

    subplot(2,1,2)
    title('Pipe section')
    xlabel('l [m]')
    hold on

    yyaxis left
    ylabel('P_{static} [Pa]')
    plot(ComponentsPipe(1,:),PStatPipe)

    yyaxis right
    ylabel('P_{dynamic} [Pa]')
    plot(ComponentsPipe(1,:),PDynPipe)
    grid on
    hold off
elseif solidity ==0
    figure(4)
    title('Main tunnel')
    xlabel('l [m]')
    hold on

    yyaxis left
    ylabel('P_{static} [Pa]')
    plot(ComponentsMain(1,:),PStatMain)

    yyaxis right
    ylabel('P_{Dynamic} [Pa]')
    plot(ComponentsMain(1,:),PDynMain)
    grid on
    hold off
end
```



Calculating frictional losses for each component of the Main Tunnel as well as the Pipe section

Theoretical frictional losses for standard components such as bends, expansions, contractions and straight sections are calculated from the "Handbook of Hydraulic Resistance" by I. E. Idelchik. Theoretical frictional losses for honeycombs are obtained from the iso-reduction curves by John L. Lumley "Reducing water-tunnel turbulence by means of a honeycomb (1967)." Theoretical frictional losses for meshes are obtained from the studies of Kurian and Fransson in "Grid-generated turbulence revisited."

These theoretical frictional losses provided an introductory basis for choosing appropriate components to minimize losses and improve the energy-ratio of the wind tunnel. Indeed, on further refinement of the design, the bends, contractions, and diffusers are expected to have smaller friction factors. It is important to note that the friction factors are calculated on the bases of the flow-rate at that component, influenced by the pump discharge and the test-section solidity.

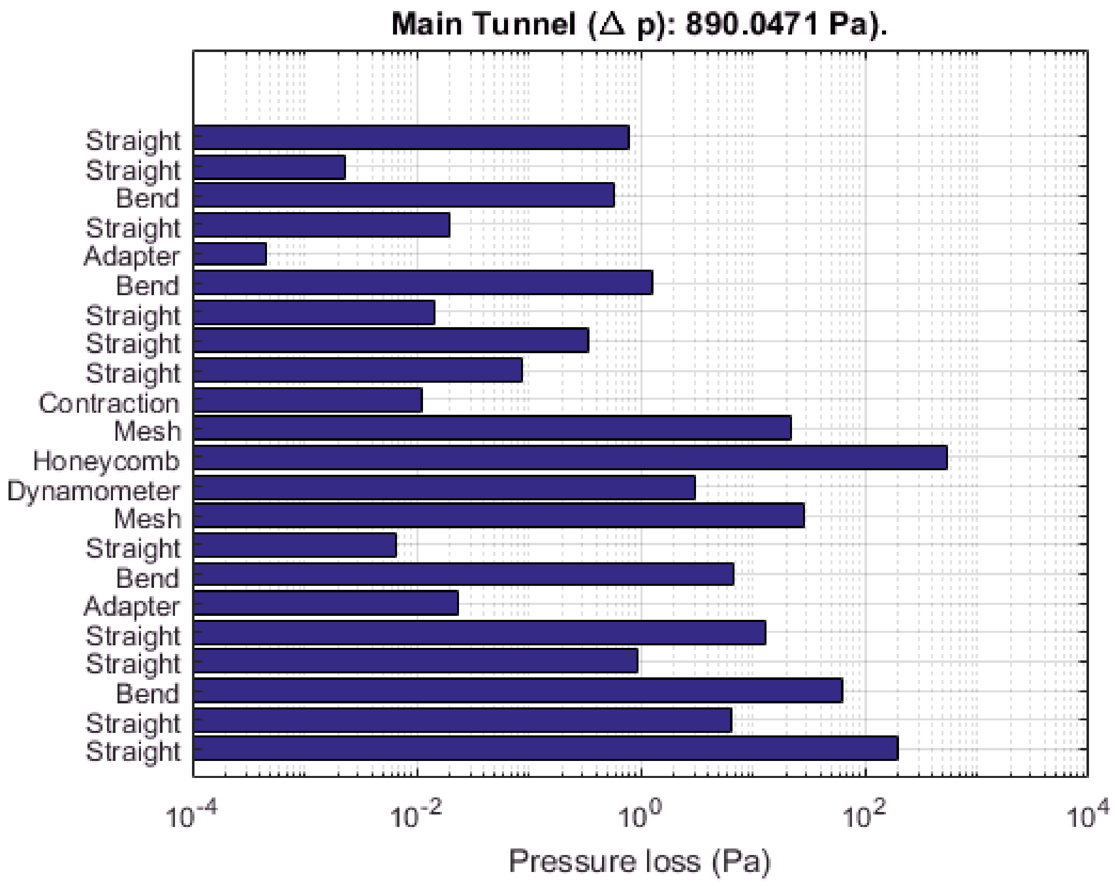
Another noteworthy point is the flow-control device for the pipe-section. Here, we assume a butterfly valve for throttling the flow in the main test-section which is taken from Diagram 9-17 of Idelchik. It is important to note that the throttling valve must be used in the range of 33% to 100% open. This allows the use of Idelchik's expressions for a square throttling valve, since observed friction factors appear to be independant of the valve shape and thickness.

```
% Applying the function friction_factor.m to estimate theoretical frictional losses
```

```
if solidity == 0     % Only the Main Tunnel is in use
    [ FricMain, QIMain ] = friction_factors( QTunnel, TunnelParams, ComponentsMain, NN, nu, QP
elseif solidity > 0
    [ FricMain, QIMain ] = friction_factors( QTunnel, TunnelParams, ComponentsMain, NN, nu, QP
    [ FricPipe, QIPipe ] = friction_factors( QPipe  , PipeParams  , ComponentsMain, NN, nu, QP
    FricPipe(end) = []; %Remove last component of FricPipe, not needed
end
```

```
Straight pipe
Straight pipe
Pipe bend
Straight pipe
Straight pipe
Circle to Square adapter
Ducted bend
Straight duct
Mesh in a duct
Dynamometer channel
Lumleys honeycomb
Mesh Reynolds number of 40005.1088 is greater than 30000. Reduce.
Mesh in a duct
Ducted contraction
Straight duct
Straight duct
Straight duct
Ducted bend
Square to circle adapter
Straight pipe
Pipe bend
Straight pipe
Straight pipe
```

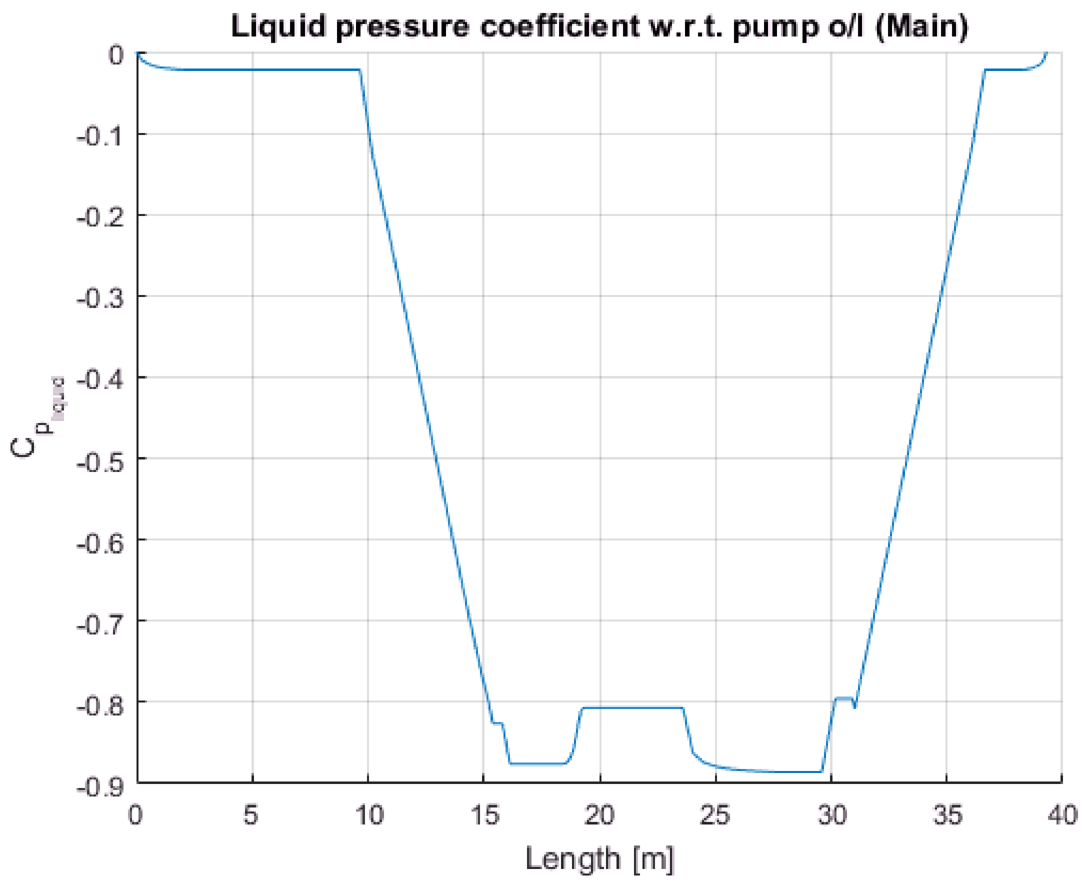
```
% Modify diffusers of the Main Tunnel to account for vanes (Reduction in \zeta to 65%). Commer
VaneCompListMain = [1,5,14,21];
for i=1:length(VaneCompListMain)
    FricMain(VaneCompListMain(i)) = 0.65*FricMain(VaneCompListMain(i));
end

% Calculating individual loss components
if solidity > 0
    for i=1:length(FricMain)-1
        LossMain(i) = FricMain(i)*(0.5 * rho * (QIMain(i)/MainArInlet(i))^2);
    end
    LossMain = [LossMain (FricMain(end) * (0.5*rho*((QPump-QPipe(1))/(0.3^2)))) ];
    for i=1:length(FricPipe)
        LossPipe(i) = FricPipe(i)*(0.5 * rho * (QIPipe(i)/PipeArInlet(i))^2);
    end

elseif solidity == 0
    for i=1:length(FricMain)
        LossMain(i) = FricMain(i)*(0.5 * rho * (QIMain(i)/MainArInlet(i))^2);
    end
end

% Plotting friction factors
if solidity > 0
    figure(6)
    grid on
    subplot(1,2,1)
    barh(LossMain)
    title(['Main Tunnel (\Delta p): ' num2str(sum(LossMain)) ' Pa).'])
    xlabel('Pressure loss (Pa)')
    set(gca, 'YTick', 1:23, 'YTickLabel', [TunnelParams(1,:) 'Throttle']);
    grid on
    set(gca,'xscale','log')

    subplot(1,2,2)
    barh(LossPipe)
    set(gca, 'YTick', 1:13, 'YTickLabel', PipeParams(1,:));
    grid on
    set(gca,'xscale','log')
    title(['Pipe Tunnel (\Delta p): ' num2str(sum(LossPipe)) ' Pa).'])
    xlabel('Pressure loss (Pa)')
elseif solidity == 0
    figure(6)
    grid on
    barh(LossMain)
    title(['Main Tunnel (\Delta p): ' num2str(sum(LossMain)) ' Pa).'])
    xlabel('Pressure loss (Pa)')
    set(gca, 'YTick', 1:22, 'YTickLabel', TunnelParams(1,:));
    grid on
    set(gca,'xscale','log')
end
```



Applying Bernoulli's theorem to calculate C_p in the main tunnel as well as the Pipe section

We do not know what the fluid pressure at the outlet to the pump is. Given this, and taking this pressure to be zero, we can calculate C_p across the tunnel length to help design a tunnel without dangerously low C_p values. Once C_p goes below zero, we expect to be concerned since the pressure in the fluid will be lower than that at the pump outlet. Similarly for the pipe section. In both cases, we non-dimensionalize with respect to the main tunnel test-section jet kinetic energy.

```
% Using Bernoulli to find the constant energy at the pump outlet
PumpOutletPressure= 0;
BernoulliConstant = (PStatMain(1) + SetP) + PDynMain(1) + PumpOutletPressure; % Here, the liqu
TestSectionJetKE  = (0.5 * rho * ((QPump - max(QPipe))/(0.3^2))^2);
% Calculating $C_p$
CPLiquidMain      = ((PStatMain + SetP) + PDynMain - BernoulliConstant)./TestSectionJetKE;
% Plotting Liquid pressure
if solidity > 0
    CPLiquidPipe      = ((PStatPipe + SetP) + PDynPipe - BernoulliConstant)./TestSectionJetKE;
    figure(7)
    subplot(2,1,1)
    hold on
    plot(ComponentsMain(1,:),CPLiquidMain)
    grid on
    title('Liquid pressure coefficient w.r.t. pump o/l (Main)')
    xlabel('Length [m]')
    ylabel('C_{p_{liquid}}')
    subplot(2,1,2)
    plot(ComponentsPipe(1,:),CPLiquidPipe)
    grid on
    title('Liquid pressure coefficient w.r.t. pump o/l (Pipe)')
    xlabel('Length [m]')
    ylabel('C_{p_{liquid}}')
elseif solidity == 0
    figure(7)
    hold on
    plot(ComponentsMain(1,:),CPLiquidMain)
    grid on
    title('Liquid pressure coefficient w.r.t. pump o/l (Main)')
    xlabel('Length [m]')
    ylabel('C_{p_{liquid}}')
end
```



Estimating Required Net Positive Suction Head, Pump Power and investigating the need of Heat Exchangers

For the net positive suction head, we require the calculation of loss head, absolute head by tunnel atmosphere, static head and the velocity head. When solidity is non-zero, the local friction factors are made global by using the jet kinetic energy of the pipe-section, while when the solidity is 0 (the throttle valve is fully open), the non-dimensionalization is with respect to the main test-section jet kinetic energy.

```
if solidity == 0
    UTs     = QPump / (0.3^2);
    H_abs   = SetP/(rho * 9.81);                                % Absolute head above the free
    H_z     = PStatMain(end)/(rho * 9.81);                      % Static pressure above the pu
    H_vp    = 3173.0724/(rho * 9.81);                           % Water Vapor Pressure at 298K
    H_v     = ((QTunnel(end)/ComponentsMain(2,end))^2)/(2*9.81);% Velocity head at pump sectio
    H_f     = (sum(FricMain))*(UTs^2)/(2 * 9.81);     % Total frictional head
    NPSH_a = H_abs + H_z - (H_f + H_v) - H_vp;                  % Available Net Positive sucti
    % Display warning for low NPSH at the pump
    if NPSH_a < 0
        disp(['NPSH available at the pump is below zero. Provide atleast ' num2str(abs(NPSH_a)
    end
elseif solidity > 0
    DiaPipe = min(ComponentsPipe(3,:));
    UTs     = QPipe(1) / (DiaPipe^2);
    H_abs   = SetP/(rho * 9.81);                                % Absolute head above the free
    H_z     = PStatMain(end)/(rho * 9.81);                      % Static pressure above the pu
    H_vp    = 3173.0724/(rho * 9.81);                           % Water Vapor Pressure at 298K
    H_v     = ((QTunnel(end)/ComponentsMain(2,end))^2)/(2*9.81);% Velocity head at pump sectio
    H_f     = (sum(FricMain) + sum(FricPipe))*(UTs^2)/(2 * 9.81);     % Total frictional head
    NPSH_a = H_abs + H_z - (H_f + H_v) - H_vp;                  % Available Net Positive sucti
    % Display warning for low NPSH at the pump
    if NPSH_a < 0
        disp(['NPSH available at the pump is below zero. Provide atleast ' num2str(abs(NPSH_a)
    end
end
```

```
NPSH available at the pump is below zero. Provide atleast 547.6064 m of static head more, upstream of th
```

With the required NPSH estimated, and with a knowledge of the theoretical frictional losses, it is possible to calculate the required pump power for a given pump efficiency.

```
if solidity == 0
    EtaPump             = 0.7;
    RequiredPumpPower   = (sum(FricMain)*(rho/2) * UTs^2) * QPump / EtaPump;             %
elseif solidity >0
    EtaPump             = 0.7;
    RequiredPumpPower   = (sum(FricMain) + sum(FricPipe))*(rho/2) * UTs^2 * QPump / EtaPump; %
end
```

Now, since we have assumed a pump efficiency, the lost power is expectedly converted to dissipative heat. By using the specific heat of the operating fluid and the tunnel capacity, one can estimate the heat capacity of the tunnel and from the dissipated heat in KW, one can find the rise in temperature of the fluid caused by the pumps.

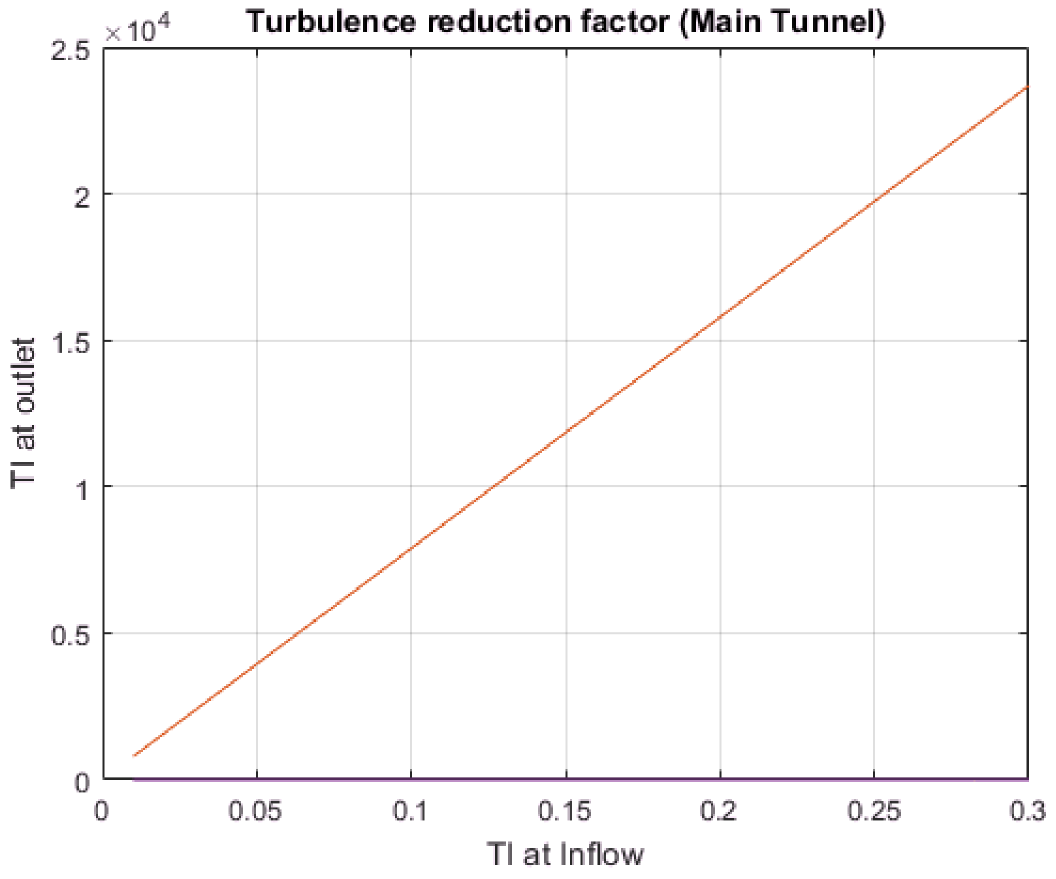
```
CpFluid = 4.186; % Specific head of operating fluid in kW/kg-sec oC
% Estimating rise in temperature every second
if solidity == 0
    TempRise   =  (sum(TunnelVolume(:,1))*rho) * CpFluid/RequiredPumpPower;             %
elseif solidity >0
    TempRise   =  ((sum(TunnelVolume(:,1)) + sum(PipeVolume(:,1)))*rho) * CpFluid/RequiredPump
end
```

Estimating turbulence reduction factors for turbulence management devices in both sections

Turbulence reduction factors are empirically calculated for each turbulence management device. Due to the high speed in the tunnel and the susceptibility of noise caused by screens, turbulence management devices are restricted to the mesh, the honeycomb and the contraction (otherwise the hydroacoustic integrity of the tunnel is severely compromised). The expressions for turbulence intensity reduction factors have been derived in theory and validated through experiments for each of the selected components. The theory is predominantly from Batchelor's rapid distortion analysis for the Contraction, Lumley's approach to estimating the honeycomb turbulence reduction factor and recent works on grid generated turbulence by Kurian and Fransson.

```
% Using the function TurbRed to estimate turbulence reduction factor for main and pipe sectio
TI_Inlet = linspace(0.01,0.30,50);    % Inlet turbulence intensity array
if solidity == 0
    RedFacMain = [];
    for i=1:length(TI_Inlet)
        [ RedFacMain(:,i) ] = TurbRed( QIMain, TunnelParams, TI_Inlet(i), NN, nu );
    end
elseif solidity >0
```

```
    RedFacMain = [];
    RedFacPipe = [];
    for i=1:length(TI_Inlet)
        [ RedFacMain(:,i) ] = TurbRed( QIMain, TunnelParams, TI_Inlet(i), NN, nu );
        [ RedFacPipe(:,i) ] = TurbRed( QIPipe, PipeParams  , TI_Inlet(i), NN, nu );
    end

end

% Plotting turbulence reduction performance

if solidity > 0
    figure(8)
    subplot(2,1,1)
    for i = 1:length(RedFacMain(:,1))
        plot(TI_Inlet,TI_Inlet.*RedFacMain(:,i))
    end
    grid on
    title('Turbulence reduction factor (Main Tunnel)')
    xlabel('TI at Inflow')
    ylabel('TI at outlet')
    subplot(2,1,2)
    for i = 1:length(RedFacPipe(:,1))
        plot(TI_Inlet,TI_Inlet.*RedFacPipe(:,i))
    end
    grid on
    title('(Pipe section)')
    xlabel('TI at Inflow')
    ylabel('TI at outlet')

elseif solidity == 0
    figure(8)
    plot(TI_Inlet,TI_Inlet.*RedFacMain)
    grid on
    title('Turbulence reduction factor (Main Tunnel)')
    xlabel('TI at Inflow')
    ylabel('TI at outlet')

end
```



Evaluating the dissolution and separation quality in the tunnel

Dynamometer calculations