1. First Approach (hydraulic diameter) code:

```
%SCRIPT TO FIND CONSISTENCY COEFFICIENT AND FLOW BEHAVIOR INDEX
USING THE POWER
%LAW MODEL%%FIRST APPROACH
clear all
close all
clc
%% Initialize variables.
filename =
'C:\Users\Hadeelatallah\Desktop\Thesis\2mm_vel4_exp1.txt';
delimiter = '\t';

%% Read columns of data as text:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%q%q%q%q%q%q%q%q%q%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate
this
% code. If an error occurs for a different file, try regenerating
the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'TextType', 'string',  'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric text to
numbers.
% Replace non-numeric text with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = mat2cell(dataArray{col},
ones(length(dataArray{col}), 1));
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4,5,6,7,8,9]
    % Converts text in the input cell array to numbers. Replaced
non-numeric
    % text with NaN.
    rawData = dataArray{col};
    for row=1:size(rawData, 1)
        % Create a regular expression to detect and remove non-
numeric prefixes and
        % suffixes.
        regexstr = '(?<prefix>.*?)(?<numbers>([-
]*(\d+[\.]*)+[\,]{0,1}\d*[eEdD]{0,1}[-+]*\d*[i]{0,1})|([-
]*(\d+[\.]*)*[\,]{1,1}\d+[eEdD]{0,1}[-
+]*\d*[i]{0,1}))(?<suffix>.*)';
        try
```

```
            result = regexp(rawData(row), regexstr, 'names');
            numbers = result.numbers;

            % Detected commas in non-thousand locations.
            invalidThousandsSeparator = false;
            if numbers.contains('.')
                thousandsRegExp = '^\d+?(\.\d{3})*\,{0,1}\d*$';
                if isempty(regexp(numbers, thousandsRegExp, 'once'))
                    numbers = NaN;
                    invalidThousandsSeparator = true;
                end
            end
            % Convert numeric text to numbers.
            if ~invalidThousandsSeparator
                numbers = strrep(numbers, '.', '');
                numbers = strrep(numbers, ',', '.');
                numbers = textscan(char(numbers), '%f');
                numericData(row, col) = numbers{1};
                raw{row, col} = numbers{1};
            end
        catch
            raw{row, col} = rawData{row};
        end
    end
end


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non-
numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

%% Allocate imported array to column variable names
abstime = cell2mat(raw(:, 1));
P1 = cell2mat(raw(:, 2));
P2 = cell2mat(raw(:, 3));
P3 = cell2mat(raw(:, 4));
P4 = cell2mat(raw(:, 5));
P5 = cell2mat(raw(:, 6));
P6 = cell2mat(raw(:, 7));
Pistonposition = cell2mat(raw(:, 8));
Absolutetime = cell2mat(raw(:, 9));


%% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans raw col
numericData rawData row regexstr result numbers
invalidThousandsSeparator thousandsRegExp R;

%%Input Parameters
L=0.105    %%distance between transducers meters
W=0.02      %%width of capillary meters
H=0.002     %%height of capillary meters
n=          %%flow index(not for zeroth iteration)
r= 0.04  %%piston radius in meters
dh=(4*H*W)/(2*(H+W));   %%hydraulic diameter
```

```
R= dh/2;       %%recalculation to radius
time = [0:0.01:34.93]';  %%variable duration of experiment

%%Ploting the pressure profiles
plot(time,P1./10,time,P2./10,time,P3./10,time,P4./10,time,P5./10,tim
e,P6./10)
grid off
legend('P1','P2','P3','P4','P5','P6');
xlabel ('Time [sec]');
ylabel ('Pressure [MPa]');

%%Find piston velocity
PistonPosition = Pistonposition./1000  %%convert piston position to
meters
plot(time,PistonPosition)
grid on
xlabel('time [sec]');
ylabel('Piston Position [m]')

v1= 0.00098153  %% variable piston velocity in m/s

%%Volumetric flow rate
A1= pi*(r^2);
A2= H*W;
Q= v1*A1;   %%volumetric flow rate through slit

%%Wall shear stress
mP2= mean(P2(2731:6391)); %%mean value of pressure in investigated
interval
mP5= mean(P5(2731:6391)); %%mean value of pressure in investigated
interval
deltaP= mP5-mP2;

tau=(deltaP*R)/(2*L); %SHEAR STRESS IN BARS

%%shear rate (newtonian)
gammadot=(4*Q)/(pi*R^3);

%% shear rate (non-newtonian)
shearrate= (Q/(pi*(R^3)))*(3+(1/n));
```

2. Second Approach (parallel plates) code:

```
%SCRIPT TO FIND CONSISTENCY COEFFICIENT AND FLOW BEHAVIOR INDEX
USING THE POWER
%LAW MODEL%%second approach
clear all
close all
clc
%% Initialize variables.
filename =
'C:\Users\Hadeelatallah\Desktop\Thesis\2mm_vel1_exp1.txt';
delimiter = '\t';

%% Read columns of data as text:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%q%q%q%q%q%q%q%q%q%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate
this
% code. If an error occurs for a different file, try regenerating
the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'TextType', 'string',  'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric text to
numbers.
% Replace non-numeric text with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = mat2cell(dataArray{col},
ones(length(dataArray{col}), 1));
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4,5,6,7,8,9]
    % Converts text in the input cell array to numbers. Replaced
non-numeric
    % text with NaN.
    rawData = dataArray{col};
    for row=1:size(rawData, 1)
        % Create a regular expression to detect and remove non-
numeric prefixes and
        % suffixes.
        regexstr = '(?<prefix>.*?)(?<numbers>([-
]*(\d+[\.]*)+[\,]{0,1}\d*[eEdD]{0,1}[-+]*\d*[i]{0,1})|([-
]*(\d+[\.]*)*[\,]{1,1}\d+[eEdD]{0,1}[-
+]*\d*[i]{0,1}))(?<suffix>.*)';
        try
```

```
            result = regexp(rawData(row), regexstr, 'names');
            numbers = result.numbers;

            % Detected commas in non-thousand locations.
            invalidThousandsSeparator = false;
            if numbers.contains('.')
                thousandsRegExp = '^\d+?(\.\d{3})*\,{0,1}\d*$';
                if isempty(regexp(numbers, thousandsRegExp, 'once'))
                    numbers = NaN;
                    invalidThousandsSeparator = true;
                end
            end
            % Convert numeric text to numbers.
            if ~invalidThousandsSeparator
                numbers = strrep(numbers, '.', '');
                numbers = strrep(numbers, ',', '.');
                numbers = textscan(char(numbers), '%f');
                numericData(row, col) = numbers{1};
                raw{row, col} = numbers{1};
            end
        catch
            raw{row, col} = rawData{row};
        end
    end
end


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non-
numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

%% Allocate imported array to column variable names
abstime = cell2mat(raw(:, 1));
P1 = cell2mat(raw(:, 2));
P2 = cell2mat(raw(:, 3));
P3 = cell2mat(raw(:, 4));
P4 = cell2mat(raw(:, 5));
P5 = cell2mat(raw(:, 6));
P6 = cell2mat(raw(:, 7));
Pistonposition = cell2mat(raw(:, 8));
Absolutetime = cell2mat(raw(:, 9));


%% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans raw col
numericData rawData row regexstr result numbers
invalidThousandsSeparator thousandsRegExp R;

%%Input Parameters
L=0.105    %%distance between transducers meters
B=0.02      %%width of capillary meters
H=0.004       %%height of capillary meters
n= 0  %%flow index(not for zeroth iteration)
r= 0.04  %%piston radius in meters
time = [0:0.01:117.80]';  %% variable duration of experiment
```

```
%%Ploting the pressure profiles
plot(time,P1./10,time,P2./10,time,P3./10,time,P4./10,time,P5./10,tim
e,P6./10)
grid off
legend('P1','P2','P3','P4','P5','P6');
xlabel ('Time [sec]');
ylabel ('Pressure [MPa]');

%%Find piston velocity
PistonPosition = Pistonposition./1000  %%convert piston position to
meters
plot(time,PistonPosition);
grid on
xlabel('time [sec]');
ylabel('Piston Position [m]')

v1= 0.00098153 %%variable piston velocity in m/s

%%Volumetric flow rate
A1= pi*(r^2);
A2= H*B;
Q= v1*A1;   %%volumetric flow rate through slit

%%Wall shear stress
deltaP= 0.028765
tau=(deltaP*H)/(2*L); %SHEAR STRESS IN BARS

%%shear rate (newtonian)
gammadot=(6*Q)/(B*H^2);

%% shear rate (non-newtonian)
shearrate= ((2*Q)/(B*(H^2)))*(2+(1/n));
```

3. Third Approach (rectangular slit) code:

```
clc
clear all
close all
%Input Parameters
L=0.105    %%distance between transducers meters
W=0.02      %%width of capillary meters
H=0.002       %%height of capillary meters

A2= H*W;    %%Cross-sectional area of capillary

a = 0.4284  %% variable geometric parameters
b = 0.92728
dh=(4*H*W)/(2*(H+W));   %%hydraulic diameter

Q= 6.772786185959756e-05    %previously calculated flow rates
(variable)
v2=Q/A2;                    %collagen velocity

%%After calculating collagen velocity, the previously calculated
shear
%%stresses (from hydraulic diameter approach) are plotted and the
curve
%%fitting tool is used to find the parameters
```