Viscosity model for particle-laden fluids (e.g. blood) up to 5 % volume fraction in micro geometry flows

Research data publication

Finn Knüppel

1. General Information

- Title: Viscosity model for particle-laden fluids (e.g. blood) up to 5% volume fraction in micro geometry flows
- Authors/Creators: Finn Knüppel
- Institutions: Institute for Turbomachinery, University of Rostock, Albert-Einstein-Straße 2, 18059 Rostock, Mecklenburg Western Pomerania, Germany

- E-Mail: finn.knueppel@uni-rotock.de

- Year of Data creation: 2023-2024

- Type of Data: Ansys 2022 R1.mat File of the raw data

- Language: English

- DOI: https://doi.org/10.18453/rosdok_id00004583

- Funding: This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 469384587

2. **Objective/Problem Statement**

We demonstrate a viscosity model for the flow simulation in narrow gaps that takes cell migration in particle laden fluids into account.

3. **Description**

This publication shows a viscosity model that accounts for cell migration in blood flow conditions for narrow gap flows and flow conditions similar to that of Ventricular Assist Devices. The groundwork for this model is derived from optical measurements in a microchannel with blood analog fluid and is validated with newly measured data using pig's blood. The data of the optical measurements is used to determine the particle distribution in the channel. With predefined parameters like the Reynolds number and the Channel Height as well as measurements for the viscosity of the fluid, the viscosity model is then able to calculate and plot the particle concentration and viscosity distribution over the channel height. Also included in this model is a step function that to simplify the distribution. Using this model in simulations showed that the Wall Shear Stress from that Simulations is closer to the values of the experiments than the simulations without this model.

4. **Data Format**

The viscosity model has a data output in the form of five different figures, and it also saves the equations generated by the model in a text document. The equations in this document can then be used in Ansys.

5. **Archive Structure**

The raw data are saved in Ansys 2022 R1.mat files, these contain the local viscosity and the local volume fraction at discrete heights in the channel.

6. **System Requirements**

   The code is written in MatLab 2020b, there are no further packages needed.

7. **Software/Script User Manuel**

   If you want to use the viscosity model, you either need the raw data that is provided in this repository or your own measurements. Furthermore, you need to adjust the code to your needs. The Reynolds number, the Channel Height, the Height of the Cell Free Layer, the particle volume fraction and the viscosities of the particle laden fluid and the carrier fluid must be known for your experiment or the chosen raw data and then changed in the code. Furthermore, you need to adjust the Measurement points for the CFL development and also measure the substance data of the fluid in the optical experiments. If you don't use the raw data from this repository, the offset in the code also needs to be adjusted. The last thing that needs to be considered is that you change the place where the text document with the equations for the local distribution is saved. The sole creator of the viscosity model is Finn Knüppel and all rights to the model lies solely with him. Neither the Institute for Turbomachinery, University of Rostock or Mr. Knüppel take responsibility for the correctness of the viscosity model.

8. Code

```
clear all
close all
%_____________________________________________________________________
%Fluid data for particle distribution
Re=100; %Reynolds number
Channel_Height=150; %Height of the channel used in the experiments
CFL_Height=15; %Height of the Cell Free Layer
particle_volume_fraction=5; %Fraction of particles in volume
mu_pa_homo=5.713; %Viscosity of the particle-laden fluid measured with a
Viscometer
mu_carrier=5.3324; %Viscosity of the carrier fluid measured with a
Viscometer

%Measurement points for CFL development
M1 = [14000, 12.5];
M2 = [20000, 15];

%Substance data of the fluid from optical measurements
mu_APTV_Fluid0w=5.3324; %Viscosity of particle free fluid
mu_APTV_Fluidphiw=5.713; %Viscosity of particle laden fluid

%Loading the results from APTV measurement
%Results can be found in the raw data of the repository
N = load('re100phi5w.mat');

%Correct the x-values, adjust counting steps to 0
Offset_x=2.5; %Offset adjustment according to measured values
N.x_temp= N.x_temp+Offset_x; %Implementation of Offset Value

%_____________________________________________________________________

%Local particle concentration at discrete heights
X=length(N.y_localvolfra_new); %Parameter for loop based on local volume
fraction from APTV measurements
m=10; %Parameter for loop

for i=[m:X-10]

   prozent(i-m+1)=N.y_localvolfra_new(i)*100; %Calculation for local
particle concentration in percent
   Channel_Height_1(i-m+1)= N.x_temp(i); %Discrete Channel Height
corresponding to local particle concentration

end

%TOP CFL-Height
Top_CFL_Height=Channel_Height-CFL_Height;

%Plot the resulting local particle concentrations at discrete heights
figure(1);
bar(Channel_Height_1,prozent,'w');
title(['Local Particle concentration over channel heigth
(Re',num2str(Re),',
',num2str(particle_volume_fraction),'w%PS)'],'FontSize',12);
xlabel('channel heigth [µm]');
ylabel( 'local particle concentration [%]');
```

```
xlim([0 150]);

%Coefficients and data for Einstein-Roscoe equation
Shear_rate=6;
particle_diameter=7.51;
nCoef=3.36*(particle_diameter^-(0.192))*Shear_rate^-(0.174);

%Loop for evaluating the polynomial function for different percentage
%values with the Einstein-Roscow Equation
for n=[m:X-10]
  Ratio_mu_mu0(n-m+1)=(1-1.35.*N.y_localvolfra_new(n)).^-nCoef;
  mu_APTV_fluid0w_dyn(n-m+1)=Ratio_mu_mu0(n-m+1)*mu_APTV_Fluid0w;
  Ratio_loc_rheo (n-m+1) =mu_APTV_fluid0w_dyn(n-m+1)/mu_APTV_Fluidphiw;
end

%Calculate values for viscosity in CFL
mu_experimental_fluid=mu_pa_homo*Ratio_loc_rheo;
Step_evaluation=Channel_Height/31;
CFL_carrier=round(CFL_Height/Step_evaluation);
number_to_replace = CFL_carrier;

 %Replacement for values in the CFL
if length(mu_experimental_fluid) >= 3 * number_to_replace
    %Replace the first and last values with the value from mu_carrier
    mu_experimental_fluid(1:number_to_replace) = mu_carrier;
       mu_experimental_fluid(end - number_to_replace + 1:end) = mu_carrier;
end

%Adjustment of the values for viscosity if rheological viscosity differs
%from global viscosity
mu_homo_loc_Distribution=mean(mu_experimental_fluid(:));

%Loop to check if viscosity distribution needs to be adjusted
if mu_homo_loc_Distribution ~= mu_pa_homo
    %Calculate the adjustment factor
    adjustment_factor = mu_pa_homo / mu_homo_loc_Distribution;

    %Adjust each value in mu_experimental_fluid
   mu_experimental_fluid_adjusted = mu_experimental_fluid *
adjustment_factor;
    mu_carrier= mu_carrier*adjustment_factor;
    %Display a message
    disp('Values have been adjusted to match mu_pa_homo.');
else
    %mu_homo_loc_Distribution is already equal to mu_pa_homo

   disp('Values are already equal to mu_pa_homo.');
end

mu_experimental_fluid = mu_experimental_fluid_adjusted;
mu_homo_loc_Distribution=mean(mu_experimental_fluid(:));

%Generating a polynomial equation for local distribution
startIndex = CFL_carrier; %Start Index for subset generation
endIndex = startIndex + (length(mu_experimental_fluid)-2*CFL_carrier); %End
Index for Subset generation

%Generation of two subsets for polynomial curve fitting
mu_experimental_fluid_subset = mu_experimental_fluid(startIndex:endIndex);
%Subset for viscosity of fluid used in measurements
```

```
Channel_Height_1_subset = Channel_Height_1(startIndex:endIndex); %Subset
for discrete Channel Heights

polyfit_degree = 4; %Degree of polynomial function for curve fitting
coefficients_poly = polyfit(Channel_Height_1_subset,
mu_experimental_fluid_subset, polyfit_degree); %Coefficients of the
polynomial function for curve fitting

syms x;
equation = coefficients_poly(1) * x^polyfit_degree; %First term of the
polynomial function

%Loop for generating the remaining terms of the polynomial function
for i = 1:polyfit_degree
    equation = equation + coefficients_poly(i+1) * x^(polyfit_degree - i);
end

equation1=vpa(equation,5); %Coefficients of the polynomial function are
rounded to five significant digits

%Display of the polynomial function as an equation for local distribution
disp(['Equation local distribution:']);
disp(['y = ', char(equation1)]);

%Fit_function to use the polynomial function for different x-values
fit_function = @(x) polyval(coefficients_poly, x);

%Generate x values for the fit curve (you can adjust this range as needed)
x_values = linspace(min(Channel_Height_1_subset),
max(Channel_Height_1_subset), (length(mu_experimental_fluid)-
2*CFL_carrier));

%Calculate the corresponding y values for the fit curve
y_fit = fit_function(x_values);

%Values of local viscosity depending on height position in the channel
value1 = mu_carrier;   %  h < CFL Height in µm
value2 = @(h) fit_function(h);   %  CFL Height in µm <= h < Channel Height
in µm - CFL Height in µm
value3 = mu_carrier;   %  h >= Channel Height in µm - CFL Height in µm

%Define the Heaviside step function (u(h))
u = @(h) double(h >= 0); %Heaviside step function

%Create the piecewise function for f(x)
f = @(h) value1 * (1 - u(h - CFL_Height)) + value2(h) .* (u(h - CFL_Height)
.* (1 - u(h - Top_CFL_Height))) + value3 * u(h - Top_CFL_Height);

%Generate an array of h values
h_values = 0:0.5:150;

%Evaluate f(x) for the given h_values
fx_values = f(h_values);

%Distribution only secondary only the CFL-Height matters
mu_experimental_fluid1=mu_experimental_fluid;

mu_experimental_fluid_loc_homo=((31*mu_pa_homo)-
(mu_carrier*6))/(length(mu_experimental_fluid)-2*CFL_carrier);
```

```
if length(mu_experimental_fluid1) >= 6
    mu_experimental_fluid1(4:end-3) = mu_experimental_fluid_loc_homo;
end

value4 = mu_carrier;   %  h < CFL Height in µm
value5 = mu_experimental_fluid_loc_homo;       %  CFL Height in µm <= h <
Channel Height in µm - CFL Height in µm
value6 = mu_carrier;   %  h >= Channel Height in µm - CFL Height in µm

%Define the Heaviside step function (u(h))
u = @(h) double(h >= 0); %Heaviside step function

%Create the piecewise function for f(x)
f = @(h) value4 * (1 - u(h - CFL_Height)) + value5 * (u(h - CFL_Height) .*
(1 - u(h - Top_CFL_Height))) + value6 * u(h - Top_CFL_Height);

%Generate an array of h values
h_values1 = 0:0.5:150; %Adjust the range as needed

%Evaluate f(x) for the given h_values
fx_values1 = f(h_values1);

%Plot the resulting step function
figure(2);
plot(h_values1, fx_values1);
xlabel('Channel height [µm]');
ylabel('f(x)');
title('Step function of figure 4 f(x)');
grid on;

%Plot of the dynamic viscosity distribution over the channel height with
local distribution
figure(3);
bar(Channel_Height_1, mu_experimental_fluid,'w');
title(['Dynamic viscosity distribution over channel heigth
(Re',num2str(Re),',
',num2str(particle_volume_fraction),'w%PS)'],'FontSize',12);
xlabel('channel heigth [µm]');
ylabel( 'local dyn. Vis concentration [mPas]');
hold on
plot(h_values, fx_values);
hold off
ylim([2.5 8]);
xlim([0 150]);

%Plot of the dynamic viscosity distribution over the channel height with
%the step model
figure(4);
bar(Channel_Height_1, mu_experimental_fluid1,'w');
title(['Dynamic viscosity distribution over channel heigth
(Re',num2str(Re),',
',num2str(particle_volume_fraction),'w%PS)'],'FontSize',12);
xlabel('channel heigth [µm]');
ylabel( 'local dyn. viscosity [mPas]');
hold on
plot(h_values1, fx_values1);
hold off
ylim([2.5 8]);
xlim([0 150]);
%_____________________________________________________________________________
```

```
%Influence of the CFL-Development

%Optical measuring points of the CFL
x = [M1(1), M2(1)];
y = [M1(2), M2(2)];

%Generating coefficients for tilted parabolic function
coefficients = polyfit(x, y.^2, 1);

%Coefficients a and b from the previous calculation
a = coefficients(1);
b = coefficients(2);
equation_text = sprintf('y^2 = %.4fx + %.4f', a, b);
disp(equation_text); %Display of the tilted parabolic function

x_values = linspace(0, M2(1), 100); %Generation of x_values between the
measuring points of the CFL

y_values = sqrt(max(0, real(a * x_values + b))); %Calculation of the
Corresponding y_values

%Check if too much y_values are 0
counter_zeros = 0;
message_displayed = false; %Flag to track whether the message has been
displayed

for i = 1:length(y_values)
    if y_values(i) == 0
        counter_zeros = counter_zeros + 1;

        if counter_zeros >= 10 && ~message_displayed
            disp('Development length CFL shorter or point not correct.');
            message_displayed = true; %Set the flag to true to indicate
that the message has been displayed
        end
    else
        counter_zeros = 0; %Reset the counter if a non-zero value is
encountered
    end
end



%Loop for calculating CFL_Development length
counter=0;
for i = 2:length(y_values)  %Start with the second value
    if y_values(i) > 1.001 * y_values(i-1)
        counter = counter + 1;
        CFL_Development_length=round((counter)/1000*10)/10;
    end

end

%Display of the Development length for the CFL
disp(['CFL_Development_length: ', num2str(CFL_Development_length), ' mm']);

%Upper CFL
y_values1 = Channel_Height-y_values;

%Plot of the CFL Development represented with a tilted parabolic
%function
```

```
figure(5)
plot(x_values, y_values, x_values, y_values1);
xlabel('CFL Development [µm]');
ylabel('Channel Height [µm] ');
title('Plot of the curve y^2 = ax + b');

%Output of the local Viscosity Distribution

%Open a text file for writing (change 'output.txt' to your desired file
name)
fileID = fopen(['Equation_local_distribution_Re', num2str(Re), '_phi ',
num2str(particle_volume_fraction), '.txt'], 'w');

%Check if the file was opened successfully
if fileID == -1
    error('Could not open the file for writing.');
end

%Define the text you want to write
Equation_local_distribution = sprintf('Equation for Ansys_CFX:
(if(x<1[micron],%s,if(x < %s [micron], if(y<(sqrt(abs(%s*x/1[micron^-1] +
(%s[micron^2])))), %s, if(y<-sqrt(abs(%s*x/1[micron^-1] + (%s[micron^2])))+
%s [micron],(%s *%s[micron]-sqrt(abs(%s *x/1[micron^-1] +
(%s[micron^2])))*%s*2)/(%s [micron]-2*sqrt(abs(%s*x/1[micron^-1] +
(%s[micron^2])))),%s)),if(x > %s [micron],if(y<%s [micron],%s ,if(y<%s
[micron],(%g)*(y/1[micron])^4+(%g)*(y/1[micron])^3+(%g)*(y/1[micron])^2+(%g
)*(y/1[micron])+(%g),%s)),0))))/1000*1[Pa]*1[s]', num2str(mu_pa_homo),
num2str(CFL_Development_length * 1000), a, b, num2str(mu_carrier), a, b,
num2str(Channel_Height), num2str(mu_pa_homo), num2str(Channel_Height), a,
b, num2str(mu_carrier), num2str(Channel_Height), a, b, num2str(mu_carrier),
num2str(CFL_Development_length * 1000), num2str(CFL_Height),
num2str(mu_carrier), num2str(Top_CFL_Height), coefficients_poly(1),
coefficients_poly(2), coefficients_poly(3), coefficients_poly(4),
coefficients_poly(5), num2str(mu_carrier));

Equation_only_STEP = sprintf('Equation for Ansys_CFX_only_STEP :
(if(x<1[micron],%s,if(x < %s [micron], if(y<(sqrt(abs(%s*x/1[micron^-1] +
(%s[micron^2])))), %s, if(y<-sqrt(abs(%s*x/1[micron^-1] + (%s[micron^2])))+
%s [micron],(%s *%s[micron]-sqrt(abs(%s *x/1[micron^-1] +
(%s[micron^2])))*%s*2)/(%s [micron]-2*sqrt(abs(%s*x/1[micron^-1] +
(%s[micron^2])))),%s)),if(x > %s [micron],if(y<%s [micron],%s ,if(y<%s
[micron],%s,%s)),0))))/1000*1[Pa]*1[s]', num2str(mu_pa_homo),
num2str(CFL_Development_length * 1000), a, b, num2str(mu_carrier), a, b,
num2str(Channel_Height), num2str(mu_pa_homo), num2str(Channel_Height), a,
b, num2str(mu_carrier), num2str(Channel_Height), a, b, num2str(mu_carrier),
num2str(CFL_Development_length * 1000), num2str(CFL_Height),
num2str(mu_carrier), num2str(Top_CFL_Height),
num2str(mu_experimental_fluid_loc_homo), num2str(mu_carrier));



%Write the text to the file
fprintf(fileID, '%s\n', Equation_local_distribution);
fprintf(fileID, '%s', Equation_only_STEP);
%Close the file
fclose(fileID);
```