**Supporting Material**

Fig. S1 shows a Surface evolver image of a 2 nL droplet as an example of the polynomial fitting method. Fig. S2 shows the residual plot of the 9th order polynomial fitted to the top half of the droplet in Fig. S1. Fig. S3 shows a 3-D model of the ULSD droplet in Fig.4 of the manuscript.

 

1. (b)

Fig. S1. a) Surface Evolver image of drop with the volume of 2 nL. b) Drop boundary using LOG method of detection.



Fig.S2. Plot of fitted 9th order polynomial to the top half of the droplet boundary and plot of residuals of the fitted polynomial which shows how well the polynomial fits the droplet edge.

An example of a modeled droplet using PFMF in MATLAB is shown in Fig.S3.



Fig.S3. 3D model of the ULSD droplet in Fig.4 using PFMF in MATLAB R2016a. The volume of the droplet is calculated to be 2.58 nL.

**Inflection Angle measurement coded in MATLAB:**

The following is the MATLAB code used to determine the inflection angle.

%% Inflection Angle Measurment

% This code measures the inflection angle of a drop on a fiber using

% polynomial fitting and image processing techniques.

%% Step 1: Load Image

clc

clear

RGB = imread('File Name');

imshow(RGB);

%% Step 2: Find the Edge of the Droplet using Laplacian of Gaussian Method(log)

I = rgb2gray(RGB);

BW = edge(I,'log',0.0001,6);

imshow(BW);

impixelinfo;

[rows,cols]=size(BW);

%% Step 3: Set up an Initial Point on the Boundary

% The bwtraceboundary routine used in step 4 requires a known single point

% on the boundary.

% edge of drop

% left side

col1 = 2;

row1 = min(find(BW(:,col1)));

%% Step 4: Trace the Boundary

%(X, Y) locations of the boundary points are found using bwtraceboundary.

%The number of points are determined experimentally to achive the minimum error.

%number of points

n=cols/2

% left side

boundary1 = bwtraceboundary(BW, [row1, col1], 'E', 8, n);

imshow(RGB); hold on;

plot(boundary1(:,2),boundary1(:,1),'g','LineWidth',3);

%% Step 5: Find the Optimum Fit

for p=3:5

 %Fit a Curve to the Edge of the Drop

 P1= polyfit(boundary1(:,2), boundary1(:,1), p);

 y1 = polyval(P1,boundary1(:,2));

 x1= boundary1(:,2);

 %Find the Inflection Angle

 %Inflection Angle can be found by calculating the value of derivative of fitted

 %polynomial at the inflection point at which the sign of

 %the curvature changes (second derivative is zero.)

 %find the values of second derivative

 D1 = polyder(P1);

 SD1= polyder(D1);

 ysd1=polyval(SD1,x1);

 %find the real roots of second derivative

 Xi1=roots(SD1);

 size\_Xi1=size(Xi1);

 imagin = imag(Xi1);

 number\_of\_real\_roots1=0;

 for i=1:size\_Xi1

 if imagin==0 & Xi1<=cols/2 & Xi1>=2

 number\_of\_real\_roots1=number\_of\_real\_roots1+1;

 xi1=Xi1(i,1);

 end

 end

 if number\_of\_real\_roots1==1

 xi1=xi1

 yi1 = polyval(P1,xi1)

 order\_of\_polynomial=p

 Polynomial=P1

 Inflection\_Angle\_Rad\_left=-atan(polyval(D1,xi1));

 Inflection\_Angle\_left=-atan(polyval(D1,xi1))\*180/pi()

 plot(x1,y1, 'Color','r','LineWidth',3);

 % Show the Value of Inflection Angle on the Plot

 plot(xi1,yi1,'+','Color','y','LineWidth',1.5);

 text(xi1+7, yi1+7, [sprintf('%1.3f',Inflection\_Angle\_left),...

 '{\circ}'],'Color','y','FontSize',14,'FontWeight','bold');

 end

end