Integrated Measurement of the Mass and Surface Charge of Discrete Microparticles Using a Suspended Microchannel Resonator

Philip Dextras[1*], Thomas P Burg[1**], Scott R Manalis[1,2***]

[1]MIT Department of Biological Engineering, [2]MIT Department of Mechanical Engineering

Supporting Information

The following document specifies the algorithms used for signal processing described in the manuscript. The document contains two source files which were compiled into executable code using MATLAB R2007a and run with the specified data files as input. Due to their large size, the data files are not provided here but can be obtained by sending a request to the corresponding author. The first file specifies the algortihm used to compute the channel wall zeta potential from the raw data obtained in the bi-directional buffer exchange experiment. The second file specifies the computation of particle mass, size, zeta potential and charge from the integrated measurement data.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Filename: wall_zeta.m
%%%
%%% Algorithm for computing the channel wall zeta potential
%%%
%%% Input: wall_zeta.mat
%%%
%%% Input Variables:
%%% [x t] (raw frequency shift data)
%%% dataset (coordinates of peaks)
%%%
%%% Output Variables:
%%% zeta_wall (mean zeta potential of 13 exchanges)
%%% zeta_wall_err (standard deviation of zeta_wall)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all

load wall_zeta.mat
x = sgolayfilt(x,1,201);

delta = t(2)-t(1);

%%% Read in the start/end time of each transition from dataset %%%

for i=1:length(dataset(:,1))
    rise_starts(i) = searchclosest(t,dataset(i,1));
    rise_ends(i) = searchclosest(t,dataset(i,2));
    fall_starts(i) = searchclosest(t,dataset(i,3));
    fall_ends(i) = searchclosest(t,dataset(i,4));
end

for i = 1:length(rise_starts)

    %%% Normalize first transition to its baseline slope %%%

    rise1 = x(rise_starts(i):rise_ends(i));
    rise1t = t(rise_starts(i):rise_ends(i));
    xfit = x(searchclosest(t,rise1t(end)-1):rise_ends(i));
    tfit = t(searchclosest(t,rise1t(end)-1):rise_ends(i));
    pfit = polyfit(tfit,xfit,1);
    rise1c = rise1 - polyval(pfit,rise1t);
    rise1c = rise1c + rise1(1) - rise1c(1);

    %%% Determine the crossing times for high & low thresholds %%%

    low = mean(rise1c(1:floor(1 + 0.1*rise1c(1))));
    high = mean(rise1c(end - floor(0.1*rise1c(end)):end));
    [null rootLr] = min(abs(rise1c-(low+0.025)));
    [null rootRr] = min(abs(rise1c-(high-0.025)));

```

```
    %%% Difference between crossing times is the transit time %%%

    dtrise(i) = rise1t(rootRr) - rise1t(rootLr);

    %%% Repeat for the second (reverse) transition %%%

    clear xfit
    clear tfit
    clear pfit
    fall1 = x(fall_starts(i):fall_ends(i));
    fall1t = t(fall_starts(i):fall_ends(i));
    xfit = x(searchclosest(t,fall1t(end)-1):fall_ends(i));
    tfit = t(searchclosest(t,fall1t(end)-1):fall_ends(i));
    pfit = polyfit(tfit,xfit,1);
    fall1c = fall1 - polyval(pfit,fall1t);
    fall1c = fall1c + fall1(1) - fall1c(1);

    low = mean(fall1c(1:floor(1 + 0.1*fall1c(1))));
    high = mean(fall1c(end - floor(0.1*fall1c(end)):end));
    [null rootLf] = min(abs(fall1c-(low-0.025)));
    [null rootRf] = min(abs(fall1c-(high+0.025)));
    dtfall(i) = fall1t(rootRf) - fall1t(rootLf);

    %%% Determine the equivalent transit time for the average velocity %%%

    dtavg(i) = 2/(1/dtfall(i) + 1/dtrise(i));

    %%% Determine the approximate time of the measurement %%%

    Tmsmt(i) = mean([t(rise_starts(i)), t(fall_ends(i))]);
end

eta = 1.003e-3;          % Viscosity of water at 20 deg C
Lchan = 828e-6;          % Total length of the channel
Lsusp = 413e-6;          % Length of the suspended part of channel
eps = 80*8.854e-12;      % Buffer permittivity
V0 = (0.3867)*20;        % Voltage applied to the channel

%%% Compute the channel zeta potential using the equivalent transit time %%%

zeta = eta*Lchan*Lsusp./(dtavg*eps*V0);
Tmsmt = (Tmsmt - Tmsmt(1));

figure
plot(Tmsmt,zeta,'k-o')
xlabel('Time (s)')
ylabel('Zeta Potential (mV)')

zeta_wall = mean(zeta)
zeta_wall_err = std(zeta)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Filename: mass_charge.m
%%%
%%% Algorithm for computing the particle mass, size, zeta potential and charge
%%%
%%% Input: mass_charge.mat
%%%
%%% Input Variables:
%%% [x t] (raw frequency shift data)
%%% peaks (coordinates of peaks)
%%%
%%% Output Variables:
%%% a (particle diameters)
%%% m (particle absolute masses)
%%% z (particle zeta potentials)
%%% q (particle charges)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear all
load mass_charge.mat

for i=1:length(peaks)
    window_starts(i) = searchclosest(t,peaks(i)-2);
    window_ends(i) = searchclosest(t,peaks(i)+1.8);
end

for m=1:length(peaks)

    peak = x(window_starts(m):window_ends(m));
    t_peak = t(window_starts(m):window_ends(m));

    %%% Interpolate raw data to convert to a constant sampling rate %%%

    delta = 1/max(peak);
    n_samples = (t_peak(end)-t_peak(1))/delta;
    t_const = linspace(t_peak(1),t_peak(end),n_samples);
    x_const = interp1(t_peak,peak,t_const);

%     figure
%     plot(t_peak,peak,'rx-',t_const,x_const,'bo')

    %%% Band-pass filter the raw data %%%

    Nyquist=0.5/delta;
    N=404;
    h1 = fir1(N,[9/Nyquist, 11/Nyquist]);
    peak_BP = circshift(conv(h1,x_const)',-(N)/2)';
    peak_BP = peak_BP(1:length(t_const));
    peak_BP = peak_BP - mean(peak_BP(500:1500));

```

```
%     figure
%     plot(t_const,x_const,'b-',t_const,peak_BP*3+1058,'r-')
%     axis([t_const(1) t_const(end) 1056.5 1058.5])

    %%% Fit the oscillatory background component of the band-passed data %%%

    n = floor(1/delta)-mod(floor(1/delta),2);
    y_bkgnd = peak_BP(n/2:3*n/2+1);
    x_bkgnd = t_const(n/2:3*n/2+1);
    beta = nlinfit(x_bkgnd,y_bkgnd,@osc,[mean(y_bkgnd),0.1,10,0]);

%     figure
%     plot(x_bkgnd,y_bkgnd,'ro',x_bkgnd,osc(beta,x_bkgnd),'b-')

    %%% Subtract the fitted background from the the band-passed data %%%

    peak_BP_norm = peak_BP-osc(beta,t_const);
    peak_BP_norm = peak_BP_norm(500:end-500);
    t_const = t_const(500:end-500);
    x_const = x_const(500:end-500);

%     figure
%     plot(t_const,x_const,'b-',t_const,peak_BP_norm*3+1058,'r-')
%     axis([t_const(1) t_const(end) 1056.5 1058.5])

    %%% Band-stop filter the raw data %%%

    h2 = fir1(N,[5/Nyquist, 15/Nyquist],'stop');
    peak_BS = circshift(conv(h2,x_const)',-(N)/2)';
    peak_BS = peak_BS(1:length(t_const));
    peak_BS = peak_BS - mean(peak_BS(n/2:3*n/2+1));
    peak_BS = sgolayfilt(peak_BS,3,201);

%     if m==36
%             figure
%             plot(t_const(500:end-500)-t_const(500),-(x_const(500:end-500)-
%                  mean(x_const(500:1000))),'k-',t_const(500:end-500)-
%                  t_const(500),-(peak_BS(500:end-500)-
%                  mean(peak_BS(500:1000))),'b')
%             axis([0 2.1 -1.2 0.2])
%             xlabel('Time (s)')
%             ylabel('Relative Frequency Shift (Hz)')
%     end

    %%% Differentiate the band-passed and band-stopped data %%%
    %%%  Modulate the derivative of the band-stopped data %%%

    ddt_peak_BS = deriv(peak_BS,delta);
    ddt_peak_BP_norm = deriv(peak_BP_norm,delta);
    beta(1) = 0;
    beta(2) = 1;
    ddt_peak_BS_mod = ddt_peak_BS.*osc(beta,t_const);
    ddt_peak_BS_mod = ddt_peak_BS_mod(500:end-500);
```

```
    ddt_peak_BP_norm = ddt_peak_BP_norm(500:end-500);
    t_const = t_const(500:end-500);

    %%% Band-pass filter the modulated derivative of the band-stopped data %%%

    ddt_peak_BS_mod_BP = circshift(conv(h1,ddt_peak_BS_mod)',-(N)/2)';
    ddt_peak_BS_mod_BP = ddt_peak_BS_mod_BP(1:length(t_const));
    ddt_peak_BS_mod_BP = ddt_peak_BS_mod_BP - mean(peak_BP(500:1500));
    denom = ddt_peak_BS_mod_BP;
    num = ddt_peak_BP_norm;

%     figure
%     plot(t_const,denom,'b-',t_const,num,'r-')

    %%% Compute the peak height and drift velocity from the band-stopped data

    peak_BS = peak_BS(500:end-500);
    peak_BS = peak_BS - mean(peak_BS(1:n/2));
    peak_BP_norm = peak_BP_norm(500:end-500);
    x_const = x_const(500:end-500);

%     figure
%     plot(t_const,peak_BS,'b-',t_const,x_const-mean(x_const(1:n/2)),'g-')

    [null midpoint] = max(peak_BS);
    f = null - mean(peak_BS(1:n/2));
    noise = 12*std(peak_BS(1:n/2));
    [null rootL] = min(abs(peak_BS(1:midpoint)-noise));
    [null rootR] = min(abs(peak_BS(midpoint+1:end)-noise));
    tL = t_const(rootL);
    tR = t_const(rootR + midpoint);
    tmid = t_const(midpoint);
    L = 413;
    deltaT(m) = tR-tL;
    vdc = L/(tR-tL);

%     if m == 36
%             figure
%             plot(t_const-t_const(1),-(denom-mean(denom(1:500))),'b',t_const-
%                  t_const(1),-(num-mean(num(1:500))),'r')
%             axis([0 2.1 -8 8])
%             xlabel('Time (s)')
%             ylabel('Relative Frequency Shift (Hz)')
%     end

    %%% Compute the spatial amplitude %%%
    %%% Start by determining its sign %%%

    clear null
    [top null] = max(num);
    Asign = sign(top/denom(null));
    num = abs(num);
    denom = abs(denom);
```

```
    omega = 2*pi*10;
    period = round(pi/(omega*delta));

    %%% Next find the local maxima of the numerator and denominator terms
    %%% for times corresponding to the particle transit  %%%

    clear null
    clear locF
    clear tlocF
    clear locenv
    clear tlocenv
    [top null] = max(num);
    start = floor(null-period/2);
    while start > 0
        start = start - period;
    end
    i=1;
    while start < length(t_const)-2*period
        start = start + period;
        [loc_num(i),j] = max(num(start:start + period));
        [loc_denom(i),k] = max(denom(start:start + period));
        tloc_num(i) = t_const(j+start);
        tloc_denom(i) = t_const(k+start);
        i=i+1;
    end

    clear null
    [Lfit,null] = searchclosest(tloc_num,tL);
    [Rfit,null] = searchclosest(tloc_num,tR);
    [Mfit,null] = searchclosest(tloc_num,tmid);
    loc_num = loc_num(Lfit:Rfit);
    tloc_num = tloc_num(Lfit:Rfit);
    loc_denom = loc_denom(Lfit:Rfit);
    tloc_denom = tloc_denom(Lfit:Rfit);

    %%% Divide the local maxima  %%%

    clear r
    clear tr
    for i=1:length(loc_num)
        r(i) = loc_num(i)/loc_denom(i);
        tr(i) = mean([tloc_num(i) tloc_denom(i)]);
    end

    r = Asign*r;
    Mfit = Mfit - Lfit + 1;
    skip = floor(length(r)/10) + 1;

%     figure
%     plot(tloc_num,loc_num,'ro',tloc_denom,loc_denom,'bo',t_const,denom,'b-',
%          t_const,num,'r-')
%     axis([tr(1) tr(end) 0 6])
%
%     figure
```

```
%     plot(tr(2:Mfit-skip),r(2:Mfit-skip),'bo-',tr(Mfit+skip:end-1),
%     r(Mfit+skip:end-1),'ro-')
%     axis([tr(1) tr(end) -6 6])

    if m == 36
        figure
        plot(tr-t_const(1),r*vdc/omega,'ko-')
        axis([0 2.1 0 70])
    end

    ratio_left = mean(r(2:Mfit-2));
    ratio_right = mean(r(Mfit+3:end-1));
    ratio_cat = cat(1,ratio_left',ratio_right')';
    ratio_error(m) = abs(std(ratio_cat)/mean(ratio_cat));

    ratio_lefts(m) = ratio_left;
    ratio_rights(m) = ratio_right;
    vdcs(m) = vdc;
    fs(m) = f;

end

%%% Determine the spatial amplitudes %%%

AL = vdcs.*ratio_lefts./omega;
AR = vdcs.*ratio_rights./omega;
A = cat(1,AR',AL')';
AT = (AR+AL)/2;

% figure
% hist(A,20)
% xlabel('amplitude (um)')

%%% Determine the particle diameters %%%

% figure
% hist(fs,15)
% axis([0.4 1.6 0 12])
% xlabel('frequency shift (Hz)')

R0 = ((3/(4*pi))*268e-15*fs/5e4).^(1/3);
a = 2*R0;

% figure
% hist(a,15)
% xlabel('diameter (um)')

%%% Determine the particle zeta potential and charge %%%

eta = 1.003e-3;                     % viscosity of water at 20 deg C
eps = 80*8.854e-12;                 % permittivity of water
zw = -0.038957693357198;            % channel wall zeta potential
```

```
E0 = 5.845236719132405e+004;        % electric field strength

% Compute particle zeta potential from spatial amplitude and wall zeta
% potential

z = eta*omega*AT*1e-6/(eps*E0)+zw;

k = 1.3806504e-23;                  % Boltzmann's constant
R = 8.314472;                       % ideal gas constant
F = 96485.3399;                     % Faraday's constant
T = 293.15;                         % Absolute temperature
e0 = 1.602176487e-19;               % fundamental charge
DL = sqrt(eps*R*T/(2*F^2*14));      % Debye length in 14mM PBS buffer
kappa = 1/DL;                       % Huckel parameter for buffer

% Compute the particle charge from particle zeta potential and diameter

q = pi*eps*kappa*(a.^2)*(k*T/e0).*(2*sinh(e0*z/(2*k*T)) +
(8./(kappa*a)).*tanh(e0*z/(4*k*T)));

clear binN
clear binP
z = 1000*z;
[N,binN] = hist(z,15);
[N,binP] = hist(z,26);
binT = cat(1,binN(1:8)',binP(15:26)')';

figure
hist(z,binT)
axis([-70 -10 0 15])
xlabel('zeta potential (mV)')
h = findobj(gca,'Type','patch');
set(h,'FaceColor','k','EdgeColor','w')

% Compute uncertainty in mean particle zeta potentials

j=1;
k=1;
for i=1:length(z)
    if z(i) < 1e3*zw
        zN(j) = z(i);
        zN_Err(j) = ratio_error(i)*(1e3*zw-zN(j));
        j = j+1;
    else
        zP(k) = z(i);
        zP_Err(k) = ratio_error(i)*(zP(k)-1e3*zw);
        k = k+1;
    end
end

zPavg = mean(zP)
zNavg = mean(zN)
zP_err = sqrt(mean(zP_Err)^2 + 1.492290825591^2)
```

```
zN_err = sqrt(mean(zN_Err)^2 + 1.492290825591^2)

%%% Determine the particle masses %%%

m=(4/3)*pi*R0.^3*(1.05e6);

figure
plot(z,m*1e12,'ko')
xlabel('zeta potential (mV)')
ylabel('mass (pg)')
axis([-70 -10 3 9])

figure
plot(q/(1e6*e0),m*1e12,'ko')
xlabel('charge (10^6xe_0)')
ylabel('mass (pg)')
axis([-2.5 0 3 9])
```