Appendix C

Matlab scripts for the L1 project

Matlab script for profile simulation for given adhesion strength

```
% **********************File name: E_minimization.m**********************
% Fan Yang
% This subroutine returns the configuration parameters with the total lowest energy value
once the adhesion strength w, reduced volume Sigma, and total area A0 are specified.

function p=E_minimization(w,Sigma,A0)

%clear all
N=6000;
lambda=linspace(0.01,50,N);
cspace=linspace(-0.99,0.99,200);

% initial total area, passed to the current subroutine
% A0=4*pi*8*8; % R0=8

%Scan lambda values and store the resulting energy E, R and c in an arrays
%c1(2), R1(2) and E1(2).
for i=1:N
   %Find the minimal value of D=peliminate([-.99,0.99],lambda(i)).
   %If min is less than zero, start root searching toward both ends
   %If min is bigger than zero, no root will be found - exit with error.
   for j=1:length(cspace)
      D(j)=peliminate_x(cspace(j),lambda(i),Sigma);
   end
   [Dmin,Imin]=min(D);
   if Dmin<0

   opts=optimset('TolX',1e-10,'TolFun',1e-10);
   %Look for root to the left and right of the minimum of D(i) to find
   %roots of D
```

```
    try
        c(i)=fzero(@(x) peliminate_x(x,lambda(i),Sigma),[-0.99,cspace(Imin)],opts);
        R3(i)=sqrt(A0/(pi*(lambda(i)*lambda(i)+2*(lambda(i)+sqrt(1-
c(i)*c(i)))^2/(1+c(i))+2*lambda(i)*acos(-c(i))+2*(1+c(i)))));
        E(i)=TotalE_w(R3(i),c(i),lambda(i),w);
    catch ME1
        idSegLast = regexp(ME1.identifier, '(?<=:)\w+$', 'match');
        if strcmp(idSegLast, 'ValuesAtEndPtsSameSign')
            c(i)=NaN;
            R3(i)=NaN;
            E(i)=NaN;
        end
    end



    else
        c(i)=NaN;
        R3(i)=NaN;
        E(i)=NaN;

    end
end

[E1min Imin]=min(E);
R3min=R3(Imin);
cmin=c(Imin);
lambda_min=lambda(Imin);

    p(1)=R3min;
    p(2)=lambda_min;
    p(3)=cmin;
```

```
% **********************File name: peliminate_x.m**********************
% Fan Yang
% "p" denotes parameterized, meaning that the user is allowed to supply a parameter
lambda here rather than a given number.
% eliminate R3 using the area and volume constraints

function D=peliminate(c0,lambda,Sigma)

if lambda>0
V=2/3*pi*(lambda/sqrt(1-c0*c0)+1)^3*(1-c0)+pi/3*(lambda+sqrt(1-
c0*c0))^2*(lambda/sqrt(1-c0*c0)+1)*(-c0)+pi*(1+c0)*lambda*lambda+pi*c0*sqrt(1-
c0*c0)*lambda+pi*(1+c0)*(1+c0)-1/3*pi*(1+c0)^3+pi*2*lambda*asin(sqrt((1+c0)/2));
A=pi*(lambda*lambda+2*(lambda+sqrt(1-c0*c0))^2/(1+c0)+2*lambda*acos(-
c0)+2*(1+c0));
%R=V/A^(3/2)-Sigma/(6*sqrt(pi));
D=V-A^(3/2)*Sigma/(6*sqrt(pi));

else D=(4/3*pi*(1-c0)+pi/3*(1-c0*c0)*(-c0)+(1+c0)^2+1/3*(1+c0)^3)/(4*pi)^1.5-
Sigma/(6*sqrt(pi));
end



% **********************File name: TotalE_w.m**********************
% Fan Yang
% This subroutine calculates and returns the total energy for an adhered vesicle when all
configuration parameters and the adhesion energy density are provided.

function TE=TotalE_w(R,c,lambda,w)

   if lambda>1,
```

```
    TE=4*pi*k*(1-c)+k*pi/sqrt(lambda^2-1)*(4*sqrt(lambda^2-1)-
2*lambda^2*atan((1+lambda*sqrt((1-c)/(1+c)))/(sqrt(lambda^2-1)))+4*c*sqrt(lambda^2-
1)+lambda^2*pi)-w*pi*lambda^2*R^2;
  elseif lambda<1,
    TE=8*pi*k+k*pi*lambda*lambda/sqrt(1-lambda*lambda)*log(((1+sqrt(1-
lambda*lambda))*(lambda+sqrt(1-c*c))/(lambda*(1-c*sqrt(1-
lambda*lambda)+lambda*sqrt(1-c*c)))))-w*pi*lambda^2*R^2;
  else
    TE=8*pi*k+2*k*pi/(1+sqrt((1-c0)/(1+c0)))-w*pi*lambda^2*R^2;
  end
```

Matlab scripts for confocal data processing

```
% Tristan Ursell - extraction of vesicle shape from confocal z-stack
% March 2009
% Vesicle Adhesion Shape Analysis
%
% Fan Yang - fitting the extracted profile to the basic shape model
clear all
close all

% file1 is a tiff stack of confocal images. There should be only one vesicle in the field of
% view. The z-stack is built such that it starts from the top of a vesicle toward the
% coverglass (adhesion zone).
[file1,aa]=imgetfile;
%[file1,dir]=uigetfile('*.tif','MultiSelect', 'on');

% get the number of images in the stack
N=length(imfinfo(file1));

% Pick an image in the middle of the stack and select a region about the center of the
% vesicle. The average intensity in the selected region is analyzed for each image, and it
```

```
% reaches maximum when the section is focused on the adhesion patch on the bottom of
% the vesicle. N is then changed to the frame number for this section so that we only
% analyze images at or above the adhesion zone.
colormap(gray);
Im=imread(file1,floor(N/2));
imagesc(Im);
% choose the region of interest by mouse clicking
[B roi]=imcrop;
for i=1:N
   Ims=imread(file1,i);
   Ims_crop=imcrop(Ims,roi);
   mean_crop(i)=mean2(Ims_crop);
end
[C bt_ind]=max(mean_crop);
N=bt_ind;

disp(['This z-stack has ' num2str(N) ' images.']);

% The center of the vescile is determined by analyzing the sections focused in the middle
% of the vesicle where a clear circle can be obtained by thresholding. The threshold is
% determined by user selecting the bright region and the dark region. Pixels with an
% intensity bigger than that of the dark region by more than 0.95*(difference between
% dark and bright region) is used in the next step for curve fitting. The circle is fitted to
% obtained the position of the center, and the cooridinates from all centering frames are
% averaged.

% choose centering frames
startf=input('Enter initial centering frame:  ');
endf=input('Enter final centering frame:  ');

centN=endf-startf+1;
```

```
%xy plane pixel conversion (um/px)
conv=0.1136;

% Cutoff between dark and light
C=0.95;

q1=input('Analyze brightness of all centering frames?(y/n) ','s');

if q1=='y'
   figure
   colormap(gray)
   for i=startf:endf
      disp('Choose a brightness cutoff...')
      disp(' ')
      Im=imread(file1,i);
      imagesc(Im)
      axis equal
      axis tight
      title('Choose dark region.')
      rect1=round(getrect);
      dark=Im(rect1(2):rect1(2)+rect1(4),rect1(1):rect1(1)+rect1(3));
      title('Choose light region.')
      rect2=round(getrect);
      light=Im(rect2(2):rect2(2)+rect2(4),rect2(1):rect2(1)+rect2(3));

      cut(i)=(mean(mean(light))-mean(mean(dark)))*C+mean(mean(dark));

      clear T
      T(:,:,1)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));
      T(:,:,2)=mat2gray(Im);
```

```
        T(:,:,3)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));

        imagesc(T)
        axis equal
        axis tight
        title(['Frame ' num2str(i) ' / ' num2str(i-startf+1) ' of ' num2str(centN)])
        pause(1)
    end
    close
else
    figure
    colormap(gray)
    disp('Choose a brightness cutoff...')
    disp(' ')
    Im=imread(file1,startf);
    imagesc(Im)
    axis equal
    axis tight
    title('Choose dark region.')
    rect1=round(getrect);
    dark=Im(rect1(2):rect1(2)+rect1(4),rect1(1):rect1(1)+rect1(3));
    title('Choose light region.')
    rect2=round(getrect);
    light=Im(rect2(2):rect2(2)+rect2(4),rect2(1):rect2(1)+rect2(3));

    cut(startf)=(mean(mean(light))-mean(mean(dark)))*C+mean(mean(dark));

    disp('Choose a brightness cutoff...')
    disp(' ')
    Im=imread(file1,endf);
    imagesc(Im)
```

```
    axis equal
    axis tight
    title('Choose dark region.')
    rect1=round(getrect);
    dark=Im(rect1(2):rect1(2)+rect1(4),rect1(1):rect1(1)+rect1(3));
    title('Choose light region.')
    rect2=round(getrect);
    light=Im(rect2(2):rect2(2)+rect2(4),rect2(1):rect2(1)+rect2(3));

    cut(endf)=(mean(mean(light))-mean(mean(dark)))*C+mean(mean(dark));

    for i=startf+1:endf-1
        cut(i)=cut(startf)+(cut(endf)-cut(startf))/centN*(i-startf);

        Im=imread(file1,i);
        clear T
        T(:,:,1)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));
        T(:,:,2)=mat2gray(Im);
        T(:,:,3)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));

        imagesc(T)
        axis equal
        axis tight
        title(['Frame ' num2str(i) ' / ' num2str(i-startf+1) ' of ' num2str(centN)])
        pause(0.5)
    end
end

%Perform the center frame circle fitting
n=1;
for i=startf:endf
```

```
Im=imread(file1,i);
[Y,X]=find(Im>cut(i));
for j=1:length(X)
   Z(j)=double(Im(Y(j),X(j)));
end

g = @(R)  sum(Z'.^n.*(R(1)-sqrt((R(2)-X).^2+(R(3)-Y).^2)).^2);

if i==startf
   R0=[size(Im,1)/4,size(Im,1)/2,size(Im,2)/2];
else
   R0=[r(i-1),X0(i-1),Y0(i-1)];
end

R=fminsearch(g,R0);

r(i)=R(1);
X0(i)=R(2);
Y0(i)=R(3);

%Plot the result
T(:,:,1)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));
T(:,:,2)=mat2gray(Im);
T(:,:,3)=mat2gray(Im)-mat2gray(Im).*double(Im>cut(i));

theta=0:0.01:2*pi;
Xp=r(i)*cos(theta)+X0(i);
Yp=r(i)*sin(theta)+Y0(i);

imagesc(T)
```

```
    hold on
    plot(Xp,Yp,'r')
    plot(X0(i),Y0(i),'bo')
    axis equal
    axis tight
    title(['Frame ' num2str(i) ' / ' num2str(i-startf+1) ' of ' num2str(centN)])
    pause(0.5)
    hold off

    clear X Y Z R T Xp Yp
end
close

Xcent=mean(X0(startf:endf));
Ycent=mean(Y0(startf:endf));

% The intensity in a shell is averaged to provide a radial intensity profile for each frame.
% determining minimum polar size
sz=size(Im);
s1=abs(sz(2)-Xcent);
s2=abs(Xcent);
s3=abs(sz(1)-Ycent);
s4=abs(Ycent);

% Establish maximum polar information radius
Rmin=round(0.9*min([s1,s2,s3,s4]));
Rminsq=Rmin^2;

% Get bin positions
clear bins
dR=0.33;
```

```
bins=conv*(dR/2:dR:Rmin);

% Create data matrices
binmean=zeros(N,length(bins));
binstd=zeros(N,length(bins));

% Find the points that lie within Rmin of the vesicle center
m=0;
for j=1:size(Im,1)
   for k=1:size(Im,2)
      if ((j-Ycent)^2+(k-Xcent)^2)<Rminsq
         m=m+1;
         R(m)=sqrt((j-Ycent)^2+(k-Xcent)^2);
         RminX(m)=k;
         RminY(m)=j;
      end
   end
end

% Performing the symmetry revolution
figure;
for i=1:N
   %i=30;
   clear V
   Im=imread(file1,i);

   for j=1:m
      V(j)=Im(RminY(j),RminX(j));
   end

   % Create histogram-averaged profile
```

```
for p=1:length(bins)
   clear binvals
   %find the points in the p-th bin
   binvals=find(and(((p-1)*dR)<R,(p*dR)>=R));
   %calculate the mean of those points
   if size(binvals,2)>0
      binmean(i,p)=mean(double(V(binvals)));
   else
      binmean(i,p)=NaN;
   end

   % Calculate the STD of these points
   if length(binvals)>1
      binstd(i,p)=std(double(V(binvals)));
   else
      binstd(i,p)=NaN;
   end

   % Plot the results
   plot(R*conv,V,'k.')
   hold on
   plot(bins,binmean(i,:),'r','LineWidth',2)
   %plot(r(i),max(binmean),'go','LineWidth',2)
   plot(bins,binmean(i,:)+binstd(i,:),'Color',[1,0.7,0],'LineWidth',1)
   plot(bins,binmean(i,:)-binstd(i,:),'Color',[1,0.7,0],'LineWidth',1)
   box('on')
   xlabel('R(um)')
   ylabel('Intensity(au)')
   title(['Frame ' num2str(i) ' of ' num2str(N)])
   hold off
end
```

```
   pause(0.1)
end
close

% Create final plots
clear Z
% step size in Z direction is 0.2 micron
dZ=0.2;
Z=0:dZ:(N-1)*dZ;
Z=Z-Z(bt_ind);
Z=-Z;
% Z coordinate is modified by taking into consideration the refractive index mismatch
% using routine test_optics.
for i=1:N
   Z(i)=10^6*test_optics(Z(i)*1e-6)+Z(i); % in microns
end

colormap(hot)

surf(bins,Z,binmean,'LineStyle','none')
hold on
surf(-bins,Z,binmean,'LineStyle','none')

view([0,90])
xlabel('R(um)')
ylabel('Z(um)')
title('Mean Polar Symmetric Intensity')
axis equal
axis tight
```

```
% Select points from the radial profile. Those higher than the average by more than
% 1.5*standard deviation is considered a point on the vesicle.
binmean_nor=mat2gray(binmean);
mean_v=mean2(binmean_nor);
std_v=std2(binmean_nor);
bg=mean_v+1.5*std_v;

X_data=[];
Z_data=[];
weight_data=[];
for i=1:N
% The first 3 points in radial profile is not used since they came from the average of too
% few pixels.
ind=find((binmean_nor(i,4:length(bins))>max(binmean_nor(i,4:length(bins)))*.85).*(bin
mean_nor(i,4:length(bins))>bg));
   if isempty(ind)
   else
      X_data=[X_data bins(ind+3)];
      for j=1:length(ind)
         Z_data=[Z_data Z(i)];
      end
      weight_data = [weight_data binmean_nor(i, ind+3)];

   end
end
weight_data = weight_data/sum(weight_data);


plot3(X_data,Z_data,sign(X_data)*1000,'b.');
plot3(-X_data,Z_data,sign(X_data)*1000,'b.');
```

```
% Now perform fitting to X_data and Z_data using basic shape model. Random initial
% conditions for the three unknown parameters are provided at the beginning of search
and the set of parameters that gives the smallest error is reported as p_final_out

fun_final=[];
p_final=[];

% Number of cycles for different random initial conditions
N_cyc=40;

for i=1:N_cyc
   % Generate four random numbers on the unit interval
   r1=rand;
   r2=rand;
   r3=rand;
   p_ini=[0.1+r1*(1-0.01) 0.1+r2*(5-0.1) -1+r3*2];
   tic
   % Search for a set of parameters that would minimize the given function
   [p_fit fun_err]=fminsearch(@(p) AdhesionError _3p(X_data,Z_data,0,p),p_ini);
   toc
   fun_final=[fun_final;fun_err];
   p_final=[p_final;p_fit];
end

[fun_final_min p_final_ind] = min(fun_final);
p_final_out = [p_final(p_final_ind, :) 0];
[x_fit z_fit] = AdhesionCurve(p_final_out);
figure;
plot(X_data, Z_data, 'rx');
hold on
plot(x_fit, z_fit, 'k.');
```

```
% ****************File name: AdhesionError_3p.m*******************
function err=AdhesionError_3p(x,z,z0,p)
% This function calculates the relative mean square error between the
% "experimental" profile (x,z) and the theoretical one given by parameter
% vector p
R3=p(1);
lambda=p(2);
c=p(3);
%x: xdata vector
%z: zdata vector
Neff=length(x);

err=0;
for i=1:1:length(x)
   %Calculate the theoretical positions by calling ModelCoor subroutine
   [xtmp ztmp ifoutside]=ModelCoor(x(i),z(i),[p z0]);
   %Calculate the errors
   err=err+((x(i)-xtmp)^2+(z(i)-ztmp)^2)/(xtmp^2+ztmp^2);
   %At the junction of two regions, ignore the data points
   Neff=Neff-ifoutside;
end
err=err/Neff;



% File name: test_optics.m
% This function calculates and returns the real z-coordinate from the nominal coordinates
% by correcting for the refractive index mismatch.

function z_max=test_optics(d)
n1=1.515;
```

```
n2=1.34;
NA=1.40;
alpha=asin(NA/n1);
theta1=linspace(0.01,alpha,200);
theta2=zeros(size(theta1));
tao_s=zeros(size(theta1));
tao_p=zeros(size(theta1));
phi_d=zeros(size(theta1));
P=ones(size(theta1));

% d=10*1e-6;
lambda=568e-9; % wavelength
k0=2*pi/lambda;
k2=2*pi*n2/lambda;
for j=1:1:200,
   theta2(j)=asin(n1*sin(theta1(j))/n2);
   tao_s(j)=2*sin(theta2(j))*cos(theta1(j))/sin(theta1(j)+theta2(j));
   tao_p(j)=2*sin(theta2(j))*cos(theta1(j))/(sin(theta1(j)+theta2(j))*cos(theta1(j)-
theta2(j)));
   phi_d(j)=-d*(n1*cos(theta1(j))-n2*cos(theta2(j)));
   %P(j)=sqrt(cos(theta1(j)));
end

z=linspace(-d/2,d/2,200);
I0=zeros(size(z));
h=zeros(size(z));
for k=1:1:200
   I0(k)=0;
   for j=1:1:200,
```

```
I0(k)=I0(k)+P(j)*sin(theta1(j))*(tao_s(j)+tao_p(j)*cos(theta2(j)))*exp(i*(k0*phi_d(j)+k2
*z(k)*cos(theta2(j))));
   end
   h(k)=abs(I0(k))*abs(I0(k));
end

%figure

%plot(z,h);

[C I]=max(h);
z_max=z(I);
```

Surface Evolver sample script

```
/* Revisions by Ken Brakke, Feb. 26, 2010
  enabled fixed area constraint
  set facet tension to 0
  fixed bug in Evolver regarding gradient of star*sq_mean_curvature on constraints.
  wrote "gogo" procedure to illustrate evolution techniques for keeping
  the bottom vertices well groomed; particularly necessary since the
   way squared mean curvature is calculated for discrete surfaces.

  Revisions by Ken Brakke, Mar. 1, 2010
    Vertices along the contact line still want to go sideways too much.
      So adding a constraint guidecon to keep the contact line vertices
      on fixed radial lines.  Had to re-center starting coordinates to
```

```
      get nice central symmetry to start with.
    Tried different versions of star sq curvature; star_normal worked
      best, winding up with no negative eigenvalues after "gogo".
*/
gravity_constant 0
/* fix area */
quantity totalarea fixed = 5 method facet_area global
/* bending energy */
// sq_mean has problems with horns.
// quantity stnsq energy modulus 1 method sq_mean_curvature global
// star_perp has trouble convergin.
//quantity stnsq energy modulus 1 method star_perp_sq_mean_curvature global
// star_normal seems to work pretty well; at least no negative eigenvalues
// after "gogo" and hessian_seek works.
quantity stnsq energy modulus 1 method star_normal_sq_mean_curvature global
// star_eff_area comes up with a few negative eigenvalues after "gogo"
// quantity stnsq energy modulus 1 method star_eff_area_sq_mean_curvature global
/* adhesion energy */
quantity adhesion energy modulus 1 method facet_scalar_integral
scalar_integrand: -5 /* user enters adhesion energy here */
/* fix the bottom on a plane */
constraint 1   /* the table top */
formula: z = 0
// Guide lines for keeping contact line vertices spaced out.
parameter guidemult = 4    // should be doubled each refinement
constraint guidecon
formula: sin(guidemult*atan2(y,x))

vertices
1  -0.5 -0.5 0.0  constraint 1,guidecon  /* 4 vertices on plane */
2   0.5 -0.5 0.0  constraint 1,guidecon
```

```
3   0.5  0.5 0.0  constraint 1,guidecon
4  -0.5  0.5 0.0  constraint 1,guidecon
5  -0.5 -0.5 1.0
6   0.5 -0.5 1.0
7   0.5  0.5 1.0
8  -0.5  0.5 1.0

edges  /* given by endpoints and attribute */
1   1 2    constraint 1,guidecon /* 4 edges on plane */
2   2 3    constraint 1,guidecon
3   3 4    constraint 1,guidecon
4   4 1    constraint 1,guidecon
5   5 6
6   6 7
7   7 8
8   8 5
9   1 5
10  2 6
11  3 7
2  4 8

faces  /* given by oriented edge loop */
1   1 10 -5  -9 density 1
2   2 11 -6 -10 density 1
3   3 12 -7 -11 density 1
4   4  9 -8 -12 density 1
5   5  6  7   8 density 1
6  -4 -3 -2  -1  color green constraint 1 density 1 adhesion

bodies  /* one body, defined by its oriented faces */
1   1 2 3 4 5 6 volume 1
```

```
//1   1 2 3 4 5 6  volume 1  density 1

read
set facet tension 0
linear_metric on // for consistently normalized eigenvalues
// Initial squish to get it started better
set vertex z z*0.7
// Grooming subroutine, for bottom facets.  Don't want to use vertex
// averaging on contact line vertices.
groom_size := 1;
groom := {
 fix vertices where on_constraint 1;
 unfix vertices vv where on_constraint 1 and sum(vv.facet, not on_constraint 1) == 0;
 refine edge where on_constraint 1 and length > groom_size;
 u; V; u; V;
 unfix vertices;
 delete edge where on_constraint 1 and length < groom_size/4;
 fix vertices vv where on_constraint 1 and sum(vv.facet, not on_constraint 1) == 0;
}

// Re-define r to automatically adjust groom_size
r :::= { guidemult *= 2; 'r'; groom_size /= 2; }

// Typical evolution.  Problem is that since curvature averages over adjacent facet area,
//rim facets on the bottom want to increase in area toward the inside, since that does not
//change the angles at the contact line vertices, but does increase the area averaged over.

gogo := {
  r;
  r;
  refine edge ee where sum(ee.facet,color==green)==1;
```

```
 m 0;  // give it a chance to adjust volume
 g;
 g;
 optimize 0.1;  // now start minimizing energy
 g;
 u;
 {g 5; groom;} 100;
 r;
 {g 5; groom; } 20;

 // try some second-order convergence
 hessian_seek;  // seems happy; hessian scale near 1.
 hessian_seek;
 v;
}
```