Rev. Colomb. Quím., 2010, **39**(3):413-425

YOUNG-LAPLACE EQUATION IN CONVENIENT POLAR COORDINATES AND ITS IMPLEMENTATION IN MATLAB®

ECUACIÓN DE YOUNG-LAPLACE EN COORDENADAS POLARES CONVENIENTES Y SU IMPLEMENTACIÓN EN MATLAB®

EQUAÇÃO DE YOUNG-LAPLACE EM COORDENADAS POLARES ADEQUADAS E SUA PROGRAMAÇÃO EM MATLAB®

Alberto R. Albis[1,2], *Adriana F. Rincón*[1]

Recibido: 18/06/10 – Aceptado: 12/11/10

ABSTRACT

A new form of expression for the Young-Laplace equation is proposed. The Young-Laplace equation is developed in a convenient polar coordinate system and programmed in MatLab®. The profile generated showed to be in agreement with those reported in literature. An algorithm that avoids profile interpolation was developed and tested for the measurement of surface tension from profiles of pendant drops.

Key words: drop profile, polar coordinates, Young-Laplace equation, MatLab®.

RESUMEN

Se propone una nueva forma para expresar la ecuación de Young-Laplace. Se desarrolló la ecuación de Young-Laplace en un sistema de coordenadas tipo polar conveniente, y su solución se programó en el *software* MatLab®. Los perfiles que se generaron mostraron una excelente coincidencia con los reportados en la literatura. Se desarrolló un algoritmo que evita la interpolación en los perfiles, el cual se evaluó para la determinación de la tensión superficial a partir de perfiles de gotas pendientes.

Palabras clave: perfil de gota, coordenadas polares, ecuación de Young-Laplace, MatLab.

RESUMO

Propõe-se uma nova forma para expressar a equação de Young-Laplace. Desenvolveu-se a equação de Young-Laplace em um sistema de coordenadas tipo polar adequado e sua solução programou-se no software MatLab®. Os perfis gerados mostraram uma excelente coincidência com os reportados na literatura. Desenvolveu-se um algoritmo que evita a inter-

1 Facultad de Ingeniería, Universidad del Atlántico, Km. 7 antigua vía a Puerto Colombia, Barranquilla, Colombia.

2 albertoalbis@mail.uiatlantico.edu.co

polação nos perfis, o qual foi avaliado para a determinação da tensão superficial a partir de perfis de gotas pingentes.

Palavras-chaves: perfil de gota, coordenadas polares, equação de Young-Laplace, MatLab.

INTRODUCTION

Surface and interfacial tension are very important properties in science and engineering due to the role they play in several processes such as emulsification and foaming. Besides, this property is very sensitive to the presence of contaminants and gives insight about the behavior of intermolecular forces at interfaces. Reliable values for this property are often required and trustfully techniques are needed to measure it.

Many techniques have been developed to measure surface tension (1). Among the commonly used methods, drop shape methods based on the analysis of the shape and size of a pendant drop has several advantages: They are absolute methods and do not depend on the contact angle between the liquid and the solid surface. Besides, the amount of sample is small; they have excellent precision, are applicable to both air-liquid and liquid–liquid interfaces, and are versatile, simple and appropriated in several situations, including extreme temperature and pressure. They have also been used to determinate the adsorption properties of biological systems, including protein adsorption at interfaces (2, 3).

The method is based on the Young-Laplace equation for capillarity. It compares the generated profile obtained by using a set of initial parameters that include the surface tension, with the experimental profile. The procedure is repeated for several values of surface tension until a good agreement between theoretically calculated and experimental profile is found (4-6).

Several changes on the original method have been introduced after its appearance in the 90's (5, 6). The modifications include improvements in the hardware (7), lens, CCD camera, light source, etc., changes in the algorithms to included new numerical methods (8-10), edge detection algorithms (11), and the application of the method to turbid samples (7, 12).

One of the disadvantages of using the Cartesian form of the Young-Laplace equation is that–for the process of profile comparison–is necessary to interpolate the generated profile (13). In this work, the Young-Laplace equation–describing the profile of pendant drops–is expressed in a convenient polar-like coordinate system to avoid interpolation. The resulting equations are solved in MatLab® software (2009, The MathWorks), and the obtained profiles are compared with data reported in the literature and with results obtained solving the Cartesian form of the equation using MatLab®.

METHODS

The Young-Laplace equation for capillarity was expressed in a convenient coordinate system, as suggested by Zholob et al. (13), and further developed by us. The dimensionless form of the equation was obtained dividing the radii length by the radii of curvature at the apex. The resulting differential equation was numerically solved using MatLab® software for different Bond numbers, and the results are compared with literature data. An algorithm was also written to obtain the Bond number from experimental profiles. The software was tested using experimental-like profiles. The experimental-like profiles were generated solving numerically the Young-Laplace equation and adding random noise to simulate real experimental profiles.

RESULTS AND DISCUSSION

Drop profile equation

The Young-Laplace equation [1] was developed to describe the profile of a meniscus inside a capillary, but the subjacent phenomena is the same that gives drops their characteristic shapes, therefore it can be used to described the profile of pendant and sessile drops (1). In this equation, the pressure difference, ΔP, is described by the two radii of curvature, R_1 and R_2, and the surface tension, γ. The pressure difference can be given in terms of the height, z, and the density difference, $\Delta\rho$, and b is the radii of curvature at the drop apex as shown in equation [2].

$$\Delta P = \gamma\left(\frac{1}{R_1}+\frac{1}{R_2}\right) \qquad [1]$$

$$\gamma\left(\frac{1}{R_1}+\frac{1}{R_2}\right) = \Delta\rho g z + \frac{2\gamma}{b} \qquad [2]$$

For symmetric shapes, the radii of curvature R_1 and R_2 can be expressed as a function of the arc length and the angle formed between the x-axis and the tangent line to the point considered, as presented in equation [3]:

$$\frac{d\phi}{d\left(s/b\right)} + \frac{\sin\phi}{x/b} = \beta\frac{z}{b} + 2 \qquad [3]$$

where β is the Bond number defined by equation [4]:

$$\beta = \frac{\Delta\rho g b^2}{\gamma} = \frac{2b^2}{a^2} \qquad [4]$$

The Bond number is positive for oblate revolution figures, i.e., bubbles below a surface and meniscus in a capillary, and negative for prolate revolution figures as pendant drops and pendant bubbles (5, 14). The dimensionless lengths are defined as shown in the following equations:

$$\bar{x} = x/b,\quad \bar{z} = z/b,\quad \bar{s} = s/b \qquad [5]$$

Taking into account geometric considerations, the following set of equations is obtained:

$$\frac{d\phi}{d\bar{s}} + \frac{\sin\phi}{\bar{x}} = \beta\bar{z} + 2 \qquad [6]$$

$$\frac{d\bar{x}}{d\bar{s}} = \cos\phi \qquad [7]$$

$$\frac{d\bar{z}}{d\bar{s}} = \sin\phi \qquad [8]$$

The initial conditions are described by equation [9]:

$$\bar{x}(0) = \bar{z}(0) = \bar{s}(0) = 0 \qquad [9]$$

Equations [6], [7] and [8] have to be solved simultaneously to obtain the calculated profile. No analytical solution has been discovered; therefore, the equations have to be solved numerically. The algorithm to solve this equation was programmed in MatLab® using buildup functions as ODE45. It is noteworthy that the integration variable in this case is the dimensionless arc length, whereas in experimental profiles the coordinates are x and z. This situation force the necessity of interpolating values in the calculated profile when we try to compare it with experimental ones, increasing the computation time and decreasing the precision of the adjustment (13).

These difficulties can be avoided by using a polar-like coordinate system as proposed by Zholob et al. (13), as shown in Figure 1.

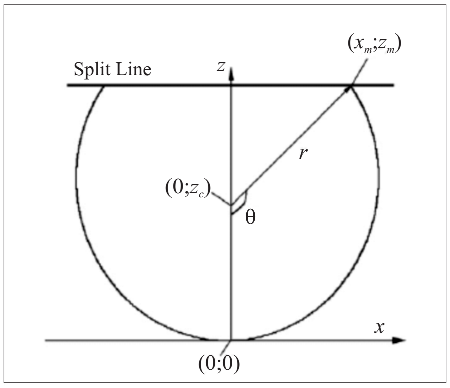

Figure 1. Coordinate System proposed by Zholob et al. to avoid data interpolation in Drop Profile Methods to measure surface tension.

The experimental coordinates are easily converted to this coordinate system and the Young-Laplace equation is transformed by using θ as the integration variable, making the comparison between experimental and calculated profiles straightforward. However, some difficulties arise when implementing this coordinate system: The position of the drop center depends on the assignment of the split line; when trying to define dimensionless magnitudes the z_c unavoidably appears in the equations; and when z_c is used to define dimensionless magnitudes, the length of the profile is immediately determined by choosing the magnitude of z_c. These difficulties make this coordinate system very awkward to be implemented. In this work, a more natural polar-like system is proposed, as shown in Figure 2.

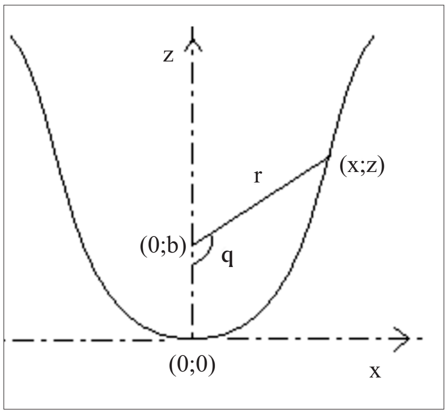

Figure 2. Coordinate System proposed in this work.

If dimensionless variables defined by equation [10] are used, the curvature and the second derivate of radii can be related by equation [11] (15):

$$\bar{r} = r/b \qquad [10]$$

$$\frac{d^2\bar{r}}{d\theta^2} = \bar{r} + \frac{1}{\bar{r}}\left(2\left(\frac{d\bar{r}}{d\theta}\right) - kQ^{3/2}\right) \qquad [11]$$

Where k is the curvature given by the Young-Laplace equation and can be expressed in the previously defined coordinate system by equations [11-14]:

$$k = \beta\left(1 - \bar{r}\cos\theta\right) + -\frac{1}{\left(1+P^2\right)^{1/2}\bar{r}\sin\theta} \qquad [12]$$

$$P = \frac{\frac{d\bar{r}}{d\theta}\sin\theta + \overline{r\cos\theta}}{\bar{r}\sin\theta - \frac{d\bar{r}}{d\theta}\cos\theta} \qquad [13]$$

$$Q = \bar{r}^2 + \left(\frac{d\bar{r}}{d\theta}\right)^2 \qquad [14]$$

With initial conditions:

$$\bar{r}(0) = 1, \quad \frac{d\bar{r}}{d\theta}(0) = 0 \qquad [15]$$

Where the boundary condition for the apex symmetry, must be satisfied:

$$\frac{d^2\bar{r}}{d\theta^2}(0) = 0 \qquad [16]$$

The solution of equation [11] to equation [16] was implemented in MatLab® and the comparison of the results with the data reported by Bashforth and Adams (14) is shown in Figure 3. Experimental and calculated profiles are superimposed. This can be seen more explicitly in Figure 4, where the differences between the radii for the Bashforth and Adams data and the results of obtained by using the equations proposed in this work are plotted. As it can be seen, the error is in the experimental uncertainty for most of the devices employed for these determinations (micrometer level).

Experimental-like profiles

Experimental-like profiles were simulated by generating theoretical profiles with the Young-Laplace equation expressed in the Cartesian system and introducing random noise to the generated profile, as equation [17]:

$$x_{dist} = x_{gen} + x_{error}, \quad z_{dist} = z_{gen} + z_{error} \qquad [17]$$

Where the subscripts *exp* and *gen* refer to the distorted and generated profile, respectively, and x_{error} and z_{error} correspond to the random distortion added to the generated profile. The magnitude of the introduced error was selected to be approximately equivalent to one pixel in a 600x400 picture and it can be positive, negative or zero for each generated point (x_{gen}, z_{gen}). The apex position was also randomly translated from the (*0, 0*) position, and a random inclination of the profile was introduced as well (15):

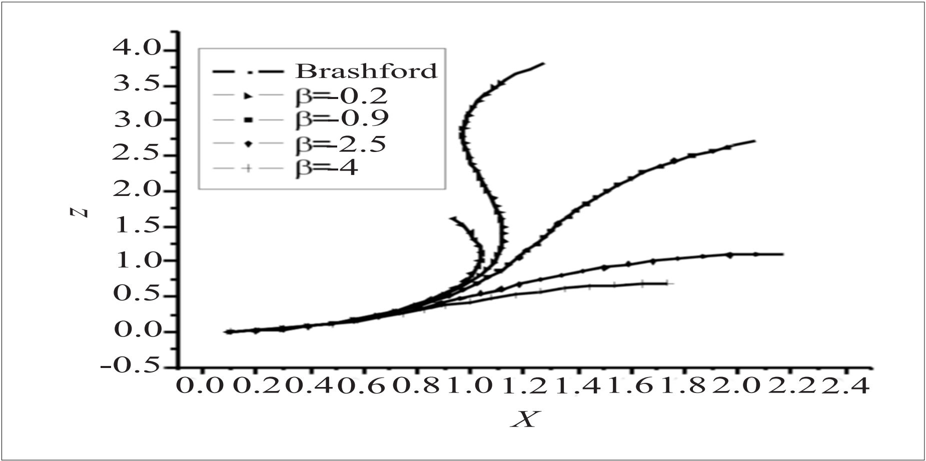

Figure 3. Comparison between literature data (solid lines) and the results obtained in this work (symbols).

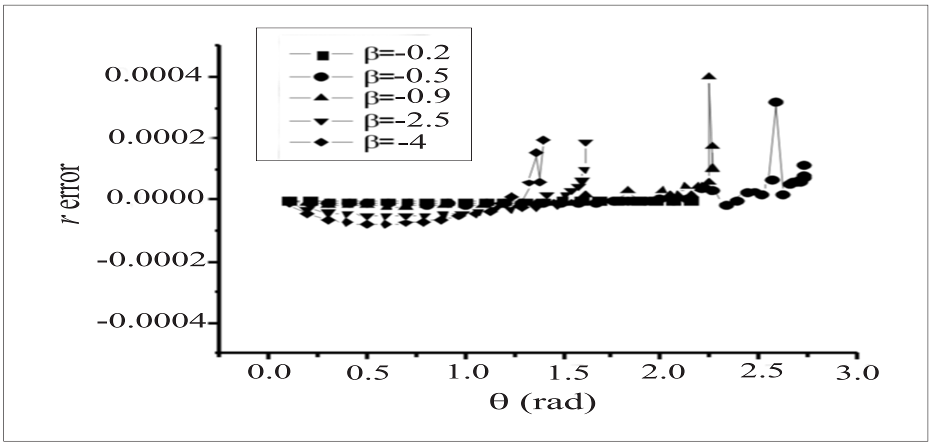

Figure 4. Absolute error in the dimensionless radii obtained by using the method proposed in this work.

$$x_{\exp} = \left(x_{gen} + x_{ap}\right)\cos\varphi - \left(z_{gen} + z_{ap}\right)\sin\varphi$$
$$z_{\exp} = \left(x_{gen} + x_{ap}\right)\sin\varphi - \left(z_{gen} + z_{ap}\right)\cos\varphi \quad [18]$$

Where the subscript *exp* refers to the coordinates of the experimental-like profile, and (x_{ap}, z_{ap}) are the new coordinates of the apex and the inclination angle of the camera is φ.

Algorithm to calculate the surface tension, and testing

An algorithm was written to evaluate the surface tension by using the method described previously. Experimental-like profiles were generated and the performance of the algorithm was then tested: a rough approximation of the apex coordinates and Bond number, β, must be supplied by the user. Using the adequate form of equation [18] the coordinates are corrected respect the apex position and camera inclination (the supposed initial value for φ is 0), and the experimental values for θ and r are calculated. The algorithm initializes the radii of curvature at the apex by adjusting several points around the apex to a circle. With these values for apex position, camera angle inclination and the parameters b and β the built in MatLab® Levenberg-Mardquardt method is initialized using as minimization function de difference between the calculated and experimental r, for each value of the experimental θ. No interpolation is necessary in this method because the calculated r can be obtained at the experimental θ values. The performance of the algorithm–by using the generated experimental-like profiles–is showed in Table 1. An excellent agreement is found between theoretical values and the values found by the algorithm written in this work. The written algorithm is presented as an m-file in Appendix.

Table 1. Performance of the written algorithm

Bond number	Fitted Bond number	Error percentage
-4.0	-3.995	-0.12
-2.5	-2.504	0.18
-0.9	-0.898	-0.20
-0.5	-0.502	0.31
-0.2	-0.199	-0.67

Another algorithm is being written to evaluate the surface tension of solutions from taken pictures or video of pendant drops. These algorithms have been widely described, but our algorithm uses the form of the Young-Laplace equation proposed here.

CONCLUSIONS

A new form of the Young-Laplace equation in polar-like coordinates was proposed. By using this method, interpolation is avoided in the calculation of the surface tension from experimental drop profiles. The implementation of the algorithm in MatLab® showed an excellent agreement between the results obtained and the results reported in the literature. Moreover, the test of the algorithm showed that the implementation in MatLab® can be used to calculate the surface tension of solutions from drop profiles.

REFERENCES

1. Adamson, A. W. and Gast, A. P. Physical Chemistry of Surfaces. New York: John Wiley & Sons, Inc. 1996. pp. 784.

2. Chen, P.; Policova, Z.; Pace-Asciak, C. R. and Neumann, A. W. Study of molecular interactions between lipids and proteins using dynamic surface tension measurements: a review. *Coll. Surf. B*. 1999. **15**: 313-324.

3. Makievski, A. V.; R.Wästneck, R.; Grigoriev, D. O.; Krägel, J. and Trukhin, D. V. Protein adsorption isotherms studied by axisymmetric drop shape analysis. *Coll. Surf. A*. 1998. **143**: 461-6.

4. Song, B.; Springer, J. Determination of interfacial tension from the profile of a pendant drop using computer-aided image processing: 2. experimental. *J. Coll. Interface Sci*. 1996. **184**: 77-91.

5. Río, O. I. d.; Neumann, A. W. axisymmetric drop shape analysis: computational methods for the measurement of interfacial properties from the shape and dimensions of pendant and sessile drops. *J. Coll. Interfac. Sci*. 1997. **196**: 136-47.

6. Song, B.; Springer, J. Determination of interfacial tension from the profile of a pendant drop using computer-aided image processing: 1. theoretical. *J. Coll. Interface Sci*. 1996. **184**: 64-76.

7. Hoorfar, M.; Neumann, A. W. Recent progress in axisymmetric drop shape analysis (ADSA). *Adv. Coll. Interfac. Sci*. 2006. **121**: 25-49.

8. Dingle, N. M.; Tjiptowidjojo, K.; Basaran, O. A.; Harris, M. T. A finite element based algorithm for determining interfacial tension (ã) from pendant drop profiles. *J. Coll. Interface Sci*. 2005. **286**: 647-60.

9. Álvarez, N. J.; Walker, L. M.; Anna, S. L. A non-gradient based algorithm for the determination of surface tension from a pendant drop: application to low Bond number drop shapes. *J. Coll. Interfac. Sci*. 2009. **333**: 557-62.

10. Thiessen, D. B.; Chione, D. J.; McCreary, C. B.; Krantz, W. B. Robust digital image analysis of pendant drop shapes. *J. Coll. Interface Sci*. 1996. **177**: 658-65.

11. Holgado-Terriza, J. A.; Gómez-Lopera, J. F.; Luque-Escamilla, P. L.; Atae-Allah, C.; Cabrerizo-Vílchez, M. A. Measurement of ultralow interfacial tension with ADSA using an entropic edge-detector. *Coll. Surf. A*. 1999. **156**: 579-86.

12. Zuo, Y. Y.; Ding, M.; Li, D. and Neumann, A. W. Further development of axisymmetric drop shape analysis-captive bubble for pulmonary surfactant related studies. *Biochem. Biophys. Acta*. 2004. **1675**: 12-20.

13. Zholob, S. A.; Makievski, A. V.; Miller, R.; Fainerman, V. B. Optimisation of calculation methods for determination of surface tensions by drop profile analysis tensiometry. *Adv. Coll. Interface Sci*. 2007. **134-135**: 322-9.

14. Bashforth, F.; Adams, J. C. An attempt to test the theories of capillary act by comparing the theoretical and measured form of drops of fluid. Cambridge, Cambridge University Press. 1883. pp. 140.

15. Polyanin, A. D.; Manzhirov, A. V. Handbook of mathematics for engineers and scientists. Boca Ratón, Chapman and Hall/CRC. 2007. pp. 1509.

APPENDIX

The written m-files are presented here. They have been tested in MatLab® 7.8.0.

```
function parcal=perfil(exp,condini)
% This function calculates a drop profi le solving numerically the Young Laplace
% equation in cartesian coordinates using the parameters supplied by the user
% in the vector exp and adds random noise to it to simulate an experimental   -like profile
%
% This generated profile is then used as an  input to test the performance
% of the software to calculate the Bond number from a simulated
% experimental profile using the Youn -Laplace equation in polar -like
% coordinates
% input:
% exp: exp is a vector that contain the parameters used to generate th  e experimental like profile
% exp is a 5 element vector:
% exp(1): Bond number
% exp(2): raddi of curvature at the apex (micrometers)
% exp(3): x coordinate of the apex
% exp(4): y coordinate of the apex
% exp(5): angle of inclination (radian)
% condini: condini is a vector that contain the inicial user estimated profile parameters
% condini(1): Bond number, (negative)
% condini(2): x apex coordinate
% condini(3): y apex coordinate
% output
% parcal: calculated parameters
% parcal(1): calculated Bond number
% parcal(2): Calculated Radii of Curvature
% (parcal(3),parcal(4)): Calculated apex position
% parcal(5): Calculated angle of inclination

if exp(1)>0
    exp(1)=-exp(1);
    disp(['The Bond number has been changed to   ' ,num2str(exp(1))])
end

perexp=perfiltipoexperimental(exp(1),exp(2),exp(3),exp(4),exp(5));   %The experimental - like profile is
generated

figure('Name' ,'Experimental-like Profile')
plot(perexp(:,1),perexp(:,2), '.','markersize' ,4)
xlabel('X / m x 10^-6')
ylabel('Y / m x 10^-6')
axis square

[parcal perfiles]=solucion(perexp,[condini(1),condini(2),condini(3)]);

figure('Name' ,'Calculated and Experimental -like profile in dimensionless form' )
plot(perfiles(:,1),perfiles(:,2), '.','markersize' ,4.5)
hold on
```

```
plot(perfiles(:,3),perfiles(:,4), '.r','markersize',4.5)
plot(-perfiles(:,3),perfiles(:,4), '.r','markersize',4.5)
xlabel('x')
ylabel('y')
hold off

disp(['Calculated Bond Number  =' ,'   ', num2str(parcal(1))])
disp(['Calculated Radii of Curvature  =' ,'   ', num2str(parcal(2))])
disp(['Calculated apex position  =' ,'   ','(',num2str(parcal(3)), ',',num2str(parcal(4)), ')'])
disp(['Calculated angle of inclination  =' ,'   ', num2str(parcal(5))])
end

function perexp=perfiltipoexperimental(bond,a,x0,y0,d)
warning off all
%it generates an experime ntal like profile adding noise to a theoretical
%profile calculated using the Young -Laplace equation in cartesian
%coordinates
%inputs:    bond: Theoretical Bond number
%           a: radii of curvature at the drop apex
%           x0: x coordinate at the  drop apex
%           y0: y coordinate at the drop apex
%           d:angle of inclination of the experimental like profile
% Outputs:
% perexp: coordinates of the experimental like profile

dx=2.5/a; % Normalized dx.
sff=-0.3226*bond^3-2.079*bond^2-2.8208*bond+3;
correalder=perfilcartadim(bond,dx,sff);
correalizq=correalder;
correalizq(:,1)=correalder(:,1)* -1;
correal=[correalizq; correalder];

[f, c]=size(correal);
clear c
facxale=rand(1,f);      % Randomization
facyale=rand(1,f);
for i=1:f
   if facxale(i)<0.33
       facxale(i)= -1;
   elseif facxale(i)<0.66
       facxale(i)=0;
   else
       facxale(i)=1;
   end

   if facyale(i)<0.33   %para el eje y
       facyale(i)= -1;
   elseif facyale(i)<0.66
       facyale(i)=0;
   else
       facyale(i)=1;
```

```
    end
end
corxdist=dx*facxale'+correal(:,1);   %adding noise to the profile
corydist=dx*facxale'+correal(:,2);
corxdist1=((corxdist)*cos(d) -(corydist)*sin(d))*a+x0;
corydist1=((corxdist)*sin(d)+(corydist)*cos(d))*a+y0;
perexp=[corxdist1,corydist1];
end

function y=perfilcartadim(bond,paso,sff)  % Young-Laplace equation in cartesian coordinates
sff=0:paso:sff;
[t y]=ode45(@younglaplaceadimensional,sff,[0 0 0]);
function dperfil=younglaplaceadimensional(s,xzfi)
%  xzfi(1)=x, xzfi(2)=z, xzfi(3)=fi
dperfil(1,1)=cos(xzfi(3));  %dx/ds= cos(fi)
dperfil(2,1)=sin(xzfi(3));  %dz/ds= sen(fi)
if s==0
    dperfil(3,1)=1;   %dfi/ds=1 at s =0
else
    dperfil(3,1)=bond*xzfi(2)+2 -sin(xzfi(3))/xzfi(1);
end
end
end

function [paropt coorcalc]=solucion(perexp,ini)
% it finds the Bond number and the coordinates of the apex of the drop for
% the experimental like profile using the initial value supplied by the
% user
% inputs:
% perexp: coordinates of the experimental profile
% ini(1)= Guessed Bond number
% ini(2)= x coordinate at the apex
% ini(3)= y coordinate at the apex
% outputs:
% paropt: paropt(1); Bond calculated
%         paropt(2): radii of curvature calculated
%         paropt(3): x coordinate at the apex calculated
%         paropt(4): y coordinate at the apex calculated
%         paropt(5): angle of inclination calculated
% coorcalc:
%            column 1: x coordinates of the experimental profile
%            column 2: y coordinates of the experimental profile
%            column 3: x coordinates calculated
%            column 4: y coordinates calculated

bondini=ini(1);
corx=perexp(:,1) -ini(2);
cory=perexp(:,2) -ini(3);
% Initializing
% calculating the initial value for the radii of curvature at the apex
rb=(corx.^2+cory.^2).^(1/2);
```

```
[rord, ix]=sort(rb);
p40=round((0.0365*bondini+0.3391)*length(corx));
p40=ix(1:p40,1);
corxb=corx(p40);
coryb=cory(p40);
corb=[corxb, coryb];  % Closest coordinates to the apex

mx = mean(corb(:,1)); my = mean(corb(:,2));   % taken from:
http://www.m athworks.com/matlabcentral/newsreader/view_thread/152405
X = corb(:,1)  - mx; Y = corb(:,2)  - my; % Get differences from means
dx2 = mean(X.^2); dy2 = mean(Y.^2);   % Get variances
t = [X,Y]\(X.^2-dx2+Y.^2-dy2)/2; % Solve least mean squares problem
a0 = t(1);  b0 = t(2);  % t is the 2 x 1 solution array [a0;b0]
bini = sqrt(dx2+dy2+a0^2+b0^2);   % Calculate the radius

opt=optimset( 'Algorithm' , 'levenberg -marquardt' , 'ScaleProblem' , 'Jacobian' );
paropt=lsqnonlin(@residuales, [bondini,bini, 0,0,0], [], [],opt,corx,cor   y);

function  res=residuales(par,x,y)
% par(1): Bond number
% par(2); radii of curvature
% par(3) x0
% par(4): y0
% par(5) inclination angle
    if par(1)>0
        par(1)=-par(1);
    end
    bond=par(1);

    xval1=(x -par(3))/par(2);  % Conversion  to dimensionless coordinates
    yval1=(y -par(4))/par(2);

    xval=xval1*cos(par(5)) -yval1*sin(par(5));
    yval=xval1*sin(par(5))+yval1*cos(par(5));

    % Changing to polar -like coordinates
    f=length(xval);
    tetaexp=zeros(f,1);
    for m=1:1: f
        if yval(m)<1
            tetaexp(m)=atan(xval(m)/(1 -yval(m)));
        else
            if xval(m)<0
                tetaexp(m)= -pi+atan(xval(m)/(1 -yval(m)));
            else
                tetaexp(m)=pi+atan(xval(m)/(1  -yval(m)));
            end
        end
    end
    rexp=((1 -yval).^2+ xval.^2).^(1/2);
    corpol=[tetaexp, rexp];
```

```
% Ordering the profile
    izq=find(corpol(:,1)<0);
    der=find(corpol(:,1)>=0);
    corpolizqx= -corpol(izq,1);
    corpolizqy=corpol(izq,2);
    corpolderx=corpol(der,1);
    corpoldery=corpol(der,2);
    val=[corpolizqx; corpolderx];
    corpolexp=[corpolizqy;corpoldery];
    [valores, m,n]=unique(val);
    corpolexp=corpolexp(m);
    a=length(valores);
    for i=a+1:1:f
        valores(i)=valores(i -1)+0.000001;
        corpolexp(i)=0;
    end
    [t rcal]=ode45(@younglappoladim,valores,[1, 0],[],bond);
    rcalc=rcal(:,1);
    rexp=corpolexp;
    a=length(rcalc);  % both vectors must have the same size to be compared
    rexp(a+1:1:f) = 0;
    rcalc(a+1:1:f)=0;
    t(a+1:1:f)=0;
    res=(abs(rcalc -rexp));
    xc=rcalc.*sin(t);
    yc=1-rcalc.*cos(t);
    coorcalc=[xval, yval, xc, yc];
end
        function dperfilp=younglappoladim(t,r,bond)   % Young Laplace equation in polar coordinates
            dperfilp=zeros(2,1);
            dperfilp(1)=r(2);  % dr/dteta
            if t==0
                dperfilp(2)=0;
            else
                Q=r(1)^2+r(2)^2;
                P=(r(2)*sin(t)+r(1)*cos(t))/(r(1)*sin(t) -r(2)*cos(t));
                k=bond*(1-r(1)*cos(t))+2-1/((1+P^2)^(1/2)*r(1)*sin(t));
                dperfilp(2)=r(1)+1/r(1)*(2*r(2)^2 -k*Q^(3/2));  %d2r/dteta2
            end
        end
end
```