MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0329072 0

ORNL/TM-10734

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

An Advanced Technique for Interfacial Tension Measurement in Liquid-Liquid Systems

M. T. Harris
C. H. Byers

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.
NTIS price codes—Printed Copy: A08 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10734

Chemical Technology Division

AN ADVANCED TECHNIQUE FOR INTERFACIAL TENSION MEASUREMENT IN LIQUID-LIQUID SYSTEMS

M. T. Harris
C. H. Byers

This report was prepared as a thesis and submitted to the faculty of the Graduate School of the University of Tennessee in partial fulfillment of the Master of Science in the Department of Chemical Engineering.

Published – November 1989

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6285
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0329072 0

CONTENTS

LIST OF FIGURES v

LIST OF TABLES vi

NOMENCLATURE vii

ABSTRACT xi

1. INTRODUCTION 1

2. LITERATURE SURVEY 7

2.1 Interfacial and Surface Tension Measurements 7

2.2 Nonlinear Parameter Estimation and Numerical Integration 10

2.3 Interfacial Tension of Various Liquid-Liquid Systems 11

3. THEORETICAL APPROACH 13

3.1 Hydrostatic Approach to Surface/Interfacial Tension 13

3.2 Intermolecular Forces and Interfacial Tension 15

3.3 Estimation of Interfacial Tension 18

3.4 Thermodynamics of Liquid Interfaces 22

3.5 The Pendant Drop Method 33

3.6 Nonlinear Parameter Estimation (NLPE) 40

3.7 Determination of Interfacial Tension by Nonlinear Parameter Estimation 48

4. EXPERIMENTAL 51

4.1 Pendant Drop Apparatus and Drop Formation 51

4.2 Video Recording and Digitization 52

4.3 Chemicals and Mixing Procedures 57

5 DISCUSSION OF RESULTS 58

5.1 Overview of Research 58

5.2 Choosing Initial Parameter Values 59

5.3 Computational Time and Accuracy 60

5.4 Interfacial Tension of Test Systems 67

5.5 Interfacial Tension of Other Systems 71
5.6 Effect of Salt on Interfacial Tension 76
5.7 Estimating Interfacial Tension 79
6. CONCLUSIONS AND RECOMMENDATIONS 82
7. REFERENCES 86
APPENDIX A: DENSITY DATA 88
APPENDIX B: EXAMPLE OF INPUT AND OUTPUT DATA SETS.... 91
APPENDIX C: COMPUTER PROGRAMS 99

LIST OF FIGURES

Figure		Page
1	Differential element of pendant drop surface.	14
2	Intermolecular forces acting at fluid-fluid interfaces.	19
3	Thermodynamic surfaces at fluid-fluid interfaces.	23
4	Effect of surfactant on surface tension.	31
5	Graphic representation of principal radii.	35
6	Alternative coordinate systems and definition of D_e and D_s.	36
7	Sum of squares surfaces for linear and nonlinear models.	45
8	Comparison between nonlinear and linearized first iteration.	46
9	Successive iteration of linearized model.	47
10	Test for perpendicularity.	49
11	Photograph of pendant drop column.	53
12	Photograph of video camera, pendant drop column and syringe dispenser.	55
13	Digitization of drop edge.	56
14	Pendant drop profile: Fordham's data (BASAD).	63
15	Pendant drop profile: Fordham's data (BASAD2T).	64
16	Pendant drop profile: Residuals (BASAD).	65
17	Pendant drop profile: Residuals (BASAD2T).	66
18	Pendant drop profile: Water-Air System ($21°C$)	69
19	Pendant drop profile: Residuals (Water-Air System).	70
20	Effect of temperature on interfacial tension.	74
21	Effect of aqueous phase salt concentration on the interfacial tension of water–2-ethyl hexanol.	78

LIST OF TABLES

Table Page

1 Test of algorithms on Fordham's data set. 62

2 Surface and interfacial tension of fluid-liquid systems. 68

3 Interfacial tension of liquid-liquid systems. 73

4 Effect of salt concentration on interfacial tension of water–2-ethyl hexanol. 77

5 Estimating interfacial tension by various methods. 80

I.1 Binary liquid density data for interfacial tension calculations. 89

I.2 Density data for water–salt–2-ethyl hexanol. 90

NOMENCLATURE

a	major axis of ellipsoid, cm
a_{cc}	capillary constant, cm
A	interfacial area, cm^2
$\mathcal{A}$	Helmholz free energy,
A_H	Hamaker constant,
b	estimates of Δ
b_p	minor axis of ellipsoid, cm
C	London constant
d_k	perpendicular distance between theoretical curve and experimental datum
D	shortest distance between atom and surface
D_e	diameter of pendant drop at equatorial plane
D_n	diameter of pendant drop for the selected plane at the vertical distance nD_e
D_s	diameter of pendant drop for the selected plane at the vertical distance D_e
e	drop eccentricity
$f(\xi, \theta)$	nonlinear model
f_r	fraction of calculated change to apply to each parameter
ΔF^a	free energy of adhesion
ΔF^c	free energy of cohesion
G	Gibbs free energy
$1/\mathcal{H}$	dimensionless shape quantity
H	enthalpy
M	molecular weight, g/mol
n	number of moles
N_A	number of atoms in a unit volume of liquid
P	pressure

ΔP	pressure difference across interface
r	radius, cm
r_a	separation distance of atoms
R_o	principal radius of curvature at the apex or origin
R_1	principal radius of curvature
R_2	principal radius of curvature
s	arc length
S	entropy
$\overline{S}$	molar entropy
$\mathcal{S}$	ratio of D_n/D_e
$S(\theta)$	true sum of squares of nonlinear model
$SS(\theta)$	sum of squares of linearized model
U	internal energy
$\overline{U}$	molar internal energy
$U(r)$	dispersion energy
V	volume
w	work of interface formation
w_1	work required to move side xx_1 away from yy_1
w_2	work required to move side xy away from x_1y_1
x	corrected horizontal coordinate of any point in a meridional section of the surface of the fluid
x_o	horizontal coordinate at the apex or origin
x_s	horizontal coordinate of digitized drop edge
x_o^M	mole fraction of organic in water phase
x_w^M	mole fraction of water in organic phase
X	dimensionless horizontal coordinate, x/R_o
X_o	dimensionless horizontal coordinate at the apex or origin, x_o/R_o
(xx_1)	length of line between x and x_1
(xy)	length of line between x and y
$(\dot{x}\dot{x}_1)$	length of line between $\dot{x}$ and $\dot{x}_1$

$(\dot{x}_1 \dot{y}_1)$	length of line between $\dot{x}_1$ and $\dot{y}_1$
y	corrected vertical coordinate at any point in a meridional section of the liquid surface
y_n	vertical distance nD_e, where n=0.8,0.9,1.0,1.1 and 1.2
y_o	vertical coordinates at the apex or origin
y_s	vertical coordinates of digitized drop edge
Y	dimensionless vertical coordinate at any point in a meridional section of the drop edge, y/R_o
Y_o	dimensionless vertical coordinate at the apex or origin, y_o/R_o
Y^E	observed dependent variable
Y'	dy/dx
Y''	d^2y/dx^2
Z	$\partial f(\xi,\theta)/\partial\theta$

Greek Letters

β	shape factor, $-\Delta\rho g R_o^2/\gamma$
β_c	shape factor,$-\Delta\rho g/\gamma$
δ	fractional change in parameter value
δ_S	fractional change in sum of squares
Δ	change in parameter estimates
γ	interfacial tension, dyne/cm = 10 μ N/cm
ϵ	residuals
μ	chemical potential
$\overline{\mu}$	molar chemical potential
θ_p	parameter estimates
ξ_k	independent variable
$\Delta\rho$	density difference between phases
τ	surface phase thickness
ϕ	slope at the point X,Y

Φ	Good's correction factor

Superscripts

α	represents bulk phase component 1
β	represents bulk phase component 2
σ	represents surface phase of interphase between bulk phases α and β

ABSTRACT

A technique for determining interfacial tension from the shape of an axisymmetric fluid-liquid interface is presented. The technique entails using a novel data acquisition method which uses a high speed video camera to record pendant drop images and a standard graphics computer to precisely measure drop edge coordinates.

The interfacial tension is determined by curve-fitting the drop edge coordinates to the Laplace equation. Curve-fitting is achieved by defining the objective function as the sum of the squared perpendicular distance between the theoretical Laplace curve and the experimental data points. An iterative nonlinear parameter estimation technique (the linearization method) is subsequently used to minimize the objective function by varying key parameters [i.e. X_o and Y_o (apex coordinates), R_o and, the shape factor, β_c]. The interfacial tension is obtained from β_c.

Experimental results show that precisions of 1.2–2.5% are achieved in determining interfacial tensions for various liquid-liquid systems. Furthermore, the effect of temperature on the interfacial tension of several systems, and the effect of salt on the interfacial tension of water–2-ethyl hexanol are presented.

AN ADVANCED TECHNIQUE FOR INTERFACIAL TENSION MEASUREMENT IN LIQUID-LIQUID SYSTEMS

M. T. Harris
C. H. Byers

1. INTRODUCTION

The interfacial tension, γ, in liquid-liquid systems and the surface tension in liquid-gas systems are probably the most fundamental equilibrium properties of an interface. Their values are one of the important factors governing the behavior of many systems of academic and practical interest (Aveyard and Briscoe 1977). For example, surface tension affects the extent of spreading when a liquid chemical is spilled on water. The interfacial tension between an organic liquid and water affects such processes as the formation of stable emulsions, the resistance to flow through orifices, and the dispersion of droplets. In liquid-liquid extraction processes, a solvent with a high interfacial tension is required to obtain good phase separation after mixing.

A prime motivation for our studying interfacial tension is the interest in advanced mass transfer operations by the Chemical Engineering Research group at the Oak Ridge National Laboratory (ORNL). The objective of these studies is the improvement of mass transfer in liquid-liquid extraction processes involving droplets in a continuous media (Wham and Byers 1985, Scott 1986). The studies have shown that the rate of mass transfer can be enhanced by forcing dispersed phase droplets to oscillate. In those studies, the imposition of a pulsed high-intensity electric field causes aqueous droplets, in a nonconducting continuous phase such as 2-ethyl-1-hexanol, to oscillate about a spherical form. Drop

eccentricity, e, a characteristic parameter for describing the deformation of the drop, is given as

$$e = \sqrt{1 - \frac{b}{a}} \tag{1}$$

where a is the major axis of the drop and b is the minor axis. A force balance between the electric forces and surface forces can relate the drop shape in terms of e to the electric field strength (Wham and Byers 1985). For small values of e, $0 \leq e \leq 0.7$

$$e \propto \frac{E_o}{\sqrt{\frac{r}{\gamma}}} \tag{2}$$

where E_o is the applied potential and r is the equivalent drop radius. Therefore, it can be seen that the primary physical properties which influence deformation of the droplet by an electric field are interfacial tension and drop size. The equation also suggests that if the interfacial tension and drop size are constant then e is proportional to the electric field strength, E_o. However, the interfacial tension may be influenced by a number of variables such as field strength or contamination of the surface of the droplet by surfactants. Therefore, the variation of interfacial tension with surfactant concentration and electric field strength is important if an accurate force balance is to be developed.

There are several methods for measuring interfacial tension. Padday (1969) and Ambwani and Fort (1979) give a list of methods such as the capillary height, Wilhelmy plate, DuNouy ring, drop weight, maximum bubble pressure, sessile drop and pendant drop. Most of the techniques, except for the maximum bubble pressure method which is difficult to operate, are suitable for measuring the surface tension of pure liquids. However, the maximum bubble pressure, Wilhelmy

plate, capillary height, Du Nouy ring, and drop weight methods are not suitable for determining the interfacial tension of liquid–liquid systems, or in some cases the surface tension of solutions, when ageing occurs. Furthermore, the capillary height, sessile drop, and pendant methods are the only methods where surface and interfacial tension values are obtained from systems where there is truly liquid–liquid and liquid–vapor equilibria.

Of the methods mentioned above, the sessile drop and pendant drop are the most generally used experimental techniques. These techniques are suitable for experimentation with (1) biological systems, (2) determination of interfacial tension between two liquids, (3) experimentation with difficult situations using reactive materials, (4) for imposing electric fields on droplets to analyze mass transfer and flow, and (5) a host of other situations.

When gravity is the only external force acting on a system, the equation which governs the shape of all pendant and sessile drops is called the Laplace equation (Padday 1969, Bashforth and Adams 1883) and is given as follows:

$$\Delta P = \gamma(\frac{1}{R_1} + \frac{1}{R_2}) \tag{3}$$

where

$$\Delta P = \text{ pressure difference across the interface,}$$
$$= -\Delta\rho g y + \frac{2\gamma}{R_o},$$
$$\Delta\rho = \text{ density difference between phases, g/cm}^3,$$
$$g = \text{ acceleration of gravity, cm/s}^2,$$
$$y = \text{ corrected vertical coordinate,}$$
$$R_o = \text{ radius of curvature at apex or origin,}$$
$$\gamma = \text{ interfacial tension, and}$$
$$R_1 \text{ and } R_2 = \text{ principal radii.}$$

This equation can also be transformed into the following dimensionless form:

$$\frac{d\phi}{ds} + \frac{sin\phi}{X} = 2 + \beta y. \tag{4}$$

where

$$\phi = \text{ slope at the point X,Y,}$$
$$s = \text{ arc length,}$$
$$X = \text{ dimensionless horizontal coordinate,}$$
$$Y = \text{ dimensionless vertical coordinate, and}$$
$$\beta = \text{ shape factor} = -\frac{\Delta\rho g R_o^2}{\gamma}.$$

Various other forms of eq. 2 and 3 have been applied to several methods for calculating interfacial tension and contact angles from pendant and sessile drops. A widely used technique developed by Andreas, Hauser, and Tucker

(1938) entails using the method of a selected plane to rapidly obtain interfacial tension from pendant and sessile drops. With their method, interfacial tension is computed from dimensionless quantities, $\mathcal{S}$ and $1/\mathcal{H}$. The parameter $\mathcal{S}$ is defined as the ratio of the measured quantities, D_s/D_e, where D_e is the diameter at the equatorial plane, and D_s is the diameter at an arbitrarily selected plane. $\mathcal{H}$ is a shape factor and is defined as

$$\mathcal{H} = \beta \left(\frac{D_e}{R_o} \right)^2 . \tag{5}$$

The interfacial tension is then computed by the following equation:

$$\gamma = \frac{\Delta\rho g (D_e)^2}{\mathcal{H}} . \tag{6}$$

Andreas, Hauser, and Tucker (1938) have generated tables of $\mathcal{S}$ as a function of $1/\mathcal{H}$. Therefore, the interfacial tension of a liquid–liquid system can be readily calculated by measuring the characteristic diameters, D_e and D_s, and applying the tables of Andreas, Hauser, and Tucker (1938).

Although, the method by Andreas, Hauser, and Tucker and other selected plane methods for determining surface and interfacial tension are simple, there are concerns whenever high accuracy and consistency are needed. A major source of error in those methods is connected with data acquisition. Since a few critical measured parameters are used to describe the whole surface of the drop, they must be determined with high precision (Andreas, Hauser, and Tucker 1938; Fordham 1948; Niederhauser and Bartell 1950; Jennings 1957; Stauffer 1965; Winkel 1965; Butler 1967; Roe, Bacchetta, and Wong 1967; Ramakrishan, Princq, and Hartland 1977; Lyons, Elbing, and Wilson 1985) . Furthermore,

the selected plane methods are statistically poor procedures.

To improve the statistical soundness of determining interfacial tension by the sessile and pendant drop methods, techniques have recently been developed to curve fit drop shape coordinate data to the Laplace equation (Maze and Burnet 1969; Maze and Burnet 1971; Vos and Los 1980; Rotenberg, Boruvka, and Neumann 1981; Girault, Schiffrin, and Smith 1982; Girault and Schriffin 1983; Huh and Reed 1983; Boryce, Schurch, Rotenberg, and Neumann 1984). These techniques are becoming more attractive as advanced video image digitizers and microcomputers are being developed.

The objective of this study is to develop a technique that allows precise measurement of interfacial tension using high speed video equipment to record surface data on the drops. Nonlinear parameter estimation methods are used to obtain interfacial tension values by curve fitting experimental drop profile data to the Laplace equation. The present method is very similar to the technique employed by Rotenberg, Boruvka, and Neumann (1981) in that the optimum interfacial tension estimates are obtained by minimizing the perpendicular distance between the experimental data points and the Laplace equation. With good initial parameter estimates, this technique will converge to a unique solution where the sum of squares is a global minimum. The present technique offers several advantages over those by Rotenberg, Boruvka, and Neumann (1981) and others. These advantages include:

(1) Good initial parameter estimates were obtained during the digitization process and therefore, convergence was faster.

(2) The algorithm converged in 5 to 10 iterations, in comparison to 25 to 50

iterations for other nonlinear parameter estimation algorithms.

(3) Since the algorithm was written to support a research project investigating the effects of electric fields on droplets, it could be easily adapted to situations where sessile and pendant drops are subjected to external fields in addition to gravitational fields, i.e. electrical fields.

(4) Readily available hardware were used for digitization and numerical computations; therefore, the design of special equipment is not required.

(5) The use of high speed video equipment allowed easy adaptation for studying the dynamic effects of contaminants on interfacial tension.

Finally, the above technique was tested on fluid–liquid systems with known interfacial tensions to determine the accuracy and precision of the method. The interfacial tension of several liquid-liquid systems (i.e., water–2-ethyl hexanol, water–octanol, glycerin–cyclohexane, and ethylene glycol–cyclohexane) that are important to electric field studies were also tested to determine the precision of the method.

2. LITERATURE SURVEY

2.1 Interfacial and Surface Tension Measurement

Several studies have used the pendant drop method for determining interfacial tension. The research has focused on solving the equations governing the drop formation and the errors associated with measuring interfacial and surface tension. Bashforth and Adams (1883) published the pioneering work of using numerical integration techniques to solve the Laplace equation of capillarity. They generated a table showing the variables in Eq. 4,

$$\frac{d\phi}{ds} + \frac{sin\phi}{X} = 2 + \beta y. \tag{4}$$

Bashforth and Adams calculated the solution to this equation for the pendant drop for values of β from -0.1 to -4.0 at intervals of 0.1, and Fordham (1948) calculated values of β ranging from -0.25 to -0.6 at intervals of 0.025. These results can be used to estimate (to an accuracy of 0.001 to 0.01%) the constants ($\frac{1}{\mathcal{H}}$ and $\mathcal{S}$) needed for measuring interfacial and surface tensions by the method suggested by Andreas, Hauser, and Tucker (1938). Fordham (1948) pointed out that a precise experimental technique is required if low percent errors in the interfacial or surface tension are desired.

Stauffer (1965) solved the Laplacian equation for capillarity by a technique involving reiterated approximations and employing a high-speed digital computer for the computations. Using that method, Stauffer was able to extend the table of $\frac{1}{\mathcal{H}}$ as a function of $\mathcal{S}$ to lower $\mathcal{S}$ values (0.197 to 0.654). However, as stated in his article, lower $\mathcal{S}$ values will yield a higher degree of error in interfacial or surface tension values because of the errors associated with measuring D_e and D_s.

Winkel (1965) proposed a method which uses the minimum diameter at the neck of the profile in place of the diameter, D_s , that was proposed by Andreas, Hauser, and Tucker (1938). Roe, Bacchetta, and Wong (1967) determined values of $\frac{1}{\mathcal{H}}$ for five different selected planes at vertical distances $y_n = nD_e$, where $n = 0.8, 0.9, 1.0, 1.1$ and 1.2, in terms of $\mathcal{S}_n = D_n/D_e$. It was shown by Ramakrishnan, Princq, and Hartland (1977) that the shape factor $\mathcal{S}_n$ is most sensitive to D_e for $n = 1.25$. Subsequently, the value of $\sqrt{\mathcal{H}}$ for the region $0.22 < \mathcal{S}_{1.25} < 0.82$ was found to be given to better than 0.5% accuracy by a

linear expression:

$$\sqrt{\mathcal{H}} = 0.724 + 0.974\mathcal{S}_{1.25}. \tag{7}$$

A recent technique has been developed by Lyons, Elbing, and Wilson (1985) that allows the determination of interfacial tension from the coordinates of any two points (excluding the origin) on the profile of a pendant or sessile interface, including pendant drops or bubbles which do not exhibit an equatorial diameter. A general method for determining the optimum choice of points that will minimize error due to definition of the drop edge was given. The authors claim that, for minimizing the error in surface tension caused by uncertainty in locating the boundary of the drop, this method is better than methods which fit a large number of profile data to the Laplace equation.

Huh and Reed (1983) developed a method of measuring very low interfacial tensions by fitting the optimum profile to 35 to 55 experimental points on the drop surface, for either sessile or pendant drops. Their algorithm required manual adjustment of the profile data to correct for asymmetry.

Rotenberg, Boruvka, and Neumann (1981) devised a novel computation scheme for determining interfacial tensions from the shapes of axisymmetric liquid-fluid interfaces. The strategy employed was to construct an objective function which expressed the error between a physically observed and a theoretical Laplacian curve. The objective function was minimized numerically using the method of incremental loading in conjunction with the Newton–Raphson method. The input information to the program consisted of the following:

1. Set of data points:$(X_n, Y_n), n = 1, \ldots, n$

2. Density difference, gravitational acceleration vector, assumed value of surface tension, X_o, Y_o, and R_o.

3. Control parameters, such as the number of repetitions of the optimization procedure, and the total number of data points.

In a more recent paper by Boyce, Schurch, Rotenberg, and Neumann (1984), measurement of surface and interfacial tension by the previously mentioned computational technique had an accuracy of the order of one percent. It was stated, however, that the attainable accuracy will be limited only by the accuracy of the digitization process.

A paper by Girault, Schiffrin, and Smith (1982) reported on a new method for the measurement of surface and interfacial tensions. The method was based on using video digitizing techniques to measure drop profile coordinates and a least squares computer fit analysis of the drop shapes. Furthermore, image acquisition required 64 ms to digitize the drop. A precision in the interfacial tension of approximately one percent was achieved.

The method by Girault, Schiffrin, and Smith (1982), and the method by Boyce, Schurch, Rotenberg, and Neumann (1984) are good; however, these techniques may require special equipment or computer software. The technique that is presented in this study uses commonly used hardware and software to achieve the same degree of precision.

2.2 Nonlinear Parameter Estimation and Numerical Integration

Determining interfacial tension by curve fitting experimental data to the Laplace equation involves laborious calculations. Furthermore, nonlinear equa-

tions are involved. This requires the use of iterative nonlinear parameter estimation routines to determine the best parameter estimates. The advent of the computer and better nonlinear parameter estimation methods is enabling such difficult problems to be solved.

An excellent discussion on the linearization technique for nonlinear parameter estimation is given by Draper and Smith (1981). Wentworth (1965) provides the mathematical model for the nonlinear parameter estimation technique that is used in the present nonlinear parameter estimation computer algorithm to determine interfacial tension. The original algorithm was developed by Whitman (1982) to solve nonlinear parameter estimation problems.

The fourth-order Runge-Kutta integration routine is based on the procedure outlined by Scarborough (1966). A method by Bridger (1977) is used to test the perpendicularity between the experimental datum and the theoretical Laplace curve.

2.3 Interfacial Tension of Various Liquid-Liquid Systems

Testing the accuracy and precision of the algorithm that is developed in this study requires interfacial tension values for several fluid–liquid systems. Furthermore, the interfacial tensions of several liquid–liquid systems that are important to electric field studies are unknown. Therefore, a literature survey was conducted to find literature values for interfacial tension, as well as to find methods for predicting interfacial tension.

A study of interfacial tension for water–organic interfaces has been done by Donahue and Bartell (1952). Their study showed a direct relationship between

interfacial tension and the degree of miscibility between the water and organic solvents. Furthermore, Antonoff's rule was shown to be valid only when applied to the liquids which form no lenses, or at most, lenses which have very low angles of contact on water. Antonoff's rule states that the interfacial tension between two liquids is the absolute difference between the surface tension of each liquid that is saturated with the other phase. Interfacial tension values between water and alcohols, alkanes, ketones, and other classes of compound were reported. The pendant drop method was employed to determine the interfacial tension values.

Good and Elbing (1970) gave a list of interfacial tension values for water–organic systems. Their paper also presented a method with the most theoretical basis for predicting interfacial tension. The method allows the prediction of interfacial tension from the surface tension of pure components and the mutual solubility of the liquids. Of particular interest to the present study, was the interfacial tension for water–cyclohexane and water–1-octanol systems.

The interfacial tension for water–2-ethyl hexanol of 13.9 dyne/cm (139 μN/cm) has been reported by Clinton (1968), and the measurements were made by the DuNouy Ring method. However, it has been reported by Padday (1969) that this method is not well suited for measuring interfacial tension for liquid-liquid systems.

Aveyard et al (1972, 1977) presented data that allows a qualitative comparison of the effect of temperature on the interfacial tension of water–alcohol systems. Girault and Schiffrin (1983) gave data for the comparison of the effect of NaCl concentration on the interfacial tension of water-2-ethyl hexanol.

3. THEORETICAL BACKGROUND

3.1 Hydrostatic Approach to Surface/Interfacial Tension

Young and Laplace (Padday 1969) developed the basic theory that associates the pressure across a curved surface to two radii of curvature. Both recognized that the attractive forces between molecules would set up a pressure across a curved liquid-fluid interface. Furthermore, if γ represents the force acting tangentially along a unit length of the surface, it is possible to derive the pressure through this surface, provided that the principle radii of curvature are known. Consider the differential element of surface shown in Fig. 1. This element may be subjected to external forces such as gravitational, and electrical fields.

If the pressure difference between the two sides is Δp, and the surface is displaced by the element dz, then the work done, w, is given by

$$w = \Delta p \times dz(xx_1)(x_1 y_1) \tag{8}$$

where

$$(xx_1)(x_1 y_1) = \text{the area of the initial element.}$$

The side xx_1 moves further away from yy_1 by the amount $(x_1 y_1)dz/R_1$, while side xy moves away from $x_1 y_1$ by the amount $(xx_1)dz/R_2$. The work required to displace each side of the element is

$$w_1 = \gamma \frac{(xx_1)(x_1 y_1)dz}{R_1}, \text{and} \tag{9}$$

ORNL DWG 87-689R2

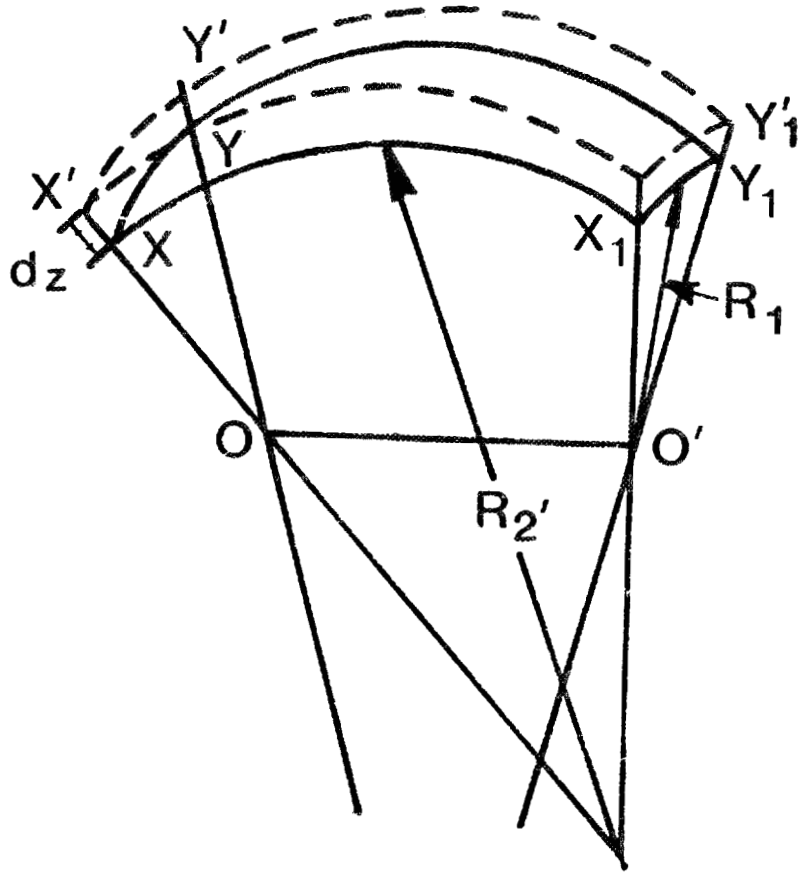

Figure 1 Differential element of pendant drop surface.

$$w_2 = \gamma \frac{(x_1 y_1)(x x_1) dz}{R_2}, \text{ respectively.} \tag{10}$$

The total work, w, is, therefore,

$$w = w_1 + w_2 = \gamma (x x_1)(x_1 y_1)(\frac{1}{R_1} + \frac{1}{R_2}) dz. \tag{11}$$

Combining Eq. 8 with Eq. 11, the following familiar expression is derived:

$$\Delta p = \gamma (\frac{1}{R_1} + \frac{1}{R_2}) \tag{12}$$

Equation 12 is Laplace's equation describing the shape of all macroscopic menisci. Other equations, such as those of Bashforth and Adams (1883), which relate the shape of the meniscus profile to the surface tension and density; and the equations used for calculating surface tension by the capillary height and the drop-weight methods are derived from the Laplace equation.

3.2 Intermolecular Forces and Interfacial Tension

As stated in the previous section, Young and Laplace recognized that intermolecular forces at an interface caused a pressure across a liquid-fluid interface. The kind of intermolecular attraction present depends on the nature of the liquids involved. Polar molecules experience Keesom (dipole-dipole) forces and Debye (dipole-induced dipole) forces of attraction. Hydrogen bonded molecules will greatly enhance the strength of attraction. Metallic bonds will be operative in liquid metals. London dispersion or Van der Waals (electrodynamic) forces are experienced in all liquids and, except for strongly polar molecules (e.g. H_2O),

those forces usually predominate over any Keesom and Debye forces which may be present.

London gave the following original expression for dispersion energy, $U(r)$, between two identical neutral atoms separated by a distance r_a, and assumed to be isotropic harmonic oscillators:

$$U(r_a) = -\frac{C}{r_a^6} \tag{13}$$

where C is the London constant.

The total attractive energy between an atom and a liquid of infinite extension and depth, may be obtained by assuming simple additivity of forces, and summing over all atom–atom interactions, expressed as $U = \sum_i U_i$,where U_i is the potential energy of interaction between the external atom and the ith atom of the liquid, assuming $U = 0$ at infinite separation. Using the potential energy function of Eq. 13, the following integral equation is given (Jaycock and Parfitt 1981):

$$U = -\int \frac{CN_A}{r_a^6} \mathrm{d}v \tag{14}$$

where

N_A = the number of atoms in a unit volume of the liquid, and

$\mathrm{d}v$ = a volume element of liquid at a distance r_a from the atom.

If N_A is assumed constant

$$U = -\frac{N_A \pi C}{6D^3} \tag{15}$$

where D is the shortest distance between atom and surface. Equations 14 and 15 are only applicable when D is much larger than the space between atoms in the interatomic spacing within the interacting plates.

The attractive potential energy U between two layers of liquid separated by a distance D in vacuum is obtained by further integration over the depth of the second layer, which leads to an energy inversely proportional to D^2

$$U = -\frac{A_H}{12\pi D^2} \tag{16}$$

where A_H is the Hamaker constant, equal to $\pi^2 N_A^2 C$. For two liquids α and β the following equation is applicable

$$U_A^{\alpha\beta} = -\frac{A_H^{\alpha\beta}}{12\pi D^2} \tag{17}$$

where

$A_H^{\alpha\beta} = \pi^2 N_\alpha N_\beta C_{\alpha\beta}$,

N_α and N_β = Number density of atoms in α and β,

$C_{\alpha\beta}$ = London Constant for the interaction of an α— atom with a β—atom.

It can be easily shown that the surface tension, γ, of a pure liquid in which only dispersion forces operate is formally given by

$$\gamma^d = \frac{A_H^{\alpha\alpha}}{24D^2}. \tag{18}$$

As stated earlier, these equations are not valid if D is of the order of the interatomic spacing within the interacting plates. Since D will always be of this order,

values obtained by this procedure should be viewed critically. This derivation (called the microscopic approach) along with the macroscopic approach (Aveyard and Vincent 1977) to surface forces, does however, provide a theoretical basis for equations used to estimate interfacial tension from the surface tension of the pure liquids and the surface tension solely due to dispersion forces.

3.3 Estimation of Interfacial Tension

Several methods have been developed to estimate interfacial tension from the surface tension of pure components, surface tension of pure component due to dispersion forces, and mutual solubility data (Grain 1982). The estimation methods with the strongest theoretical base have been described by Fowkes (1964), Good and Elbing (1970), and Aveyard and Vincent (1977). In these derivations, it is expected that, based on the theory of solubility of nonelectrolytes and the Berthelot "geometric mean" hypothesis for the attractive constant in the van der Waals equation (Girifalco and Good, 1957), the free energy of adhesion between two phases might be given by the geometric mean of the free energies of cohesion of the separate phases (Fig. 2):

$$|\Delta F_{12}^{a}| = \sqrt{\Delta F_1^c \, \Delta F_2^c}. \tag{19}$$

It has furthermore been shown by Girifalco and Good (1957) that the free energy of cohesion and adhesion, ΔF_{12}^{a} and ΔF_i^c are equal to the total energy of cohesion and adhesion, U and $U_A^{\alpha\beta}$. Therefore, noting Eqs. 17 and 18, Eq. 19 may be transformed to

ORNL DWG 87-677

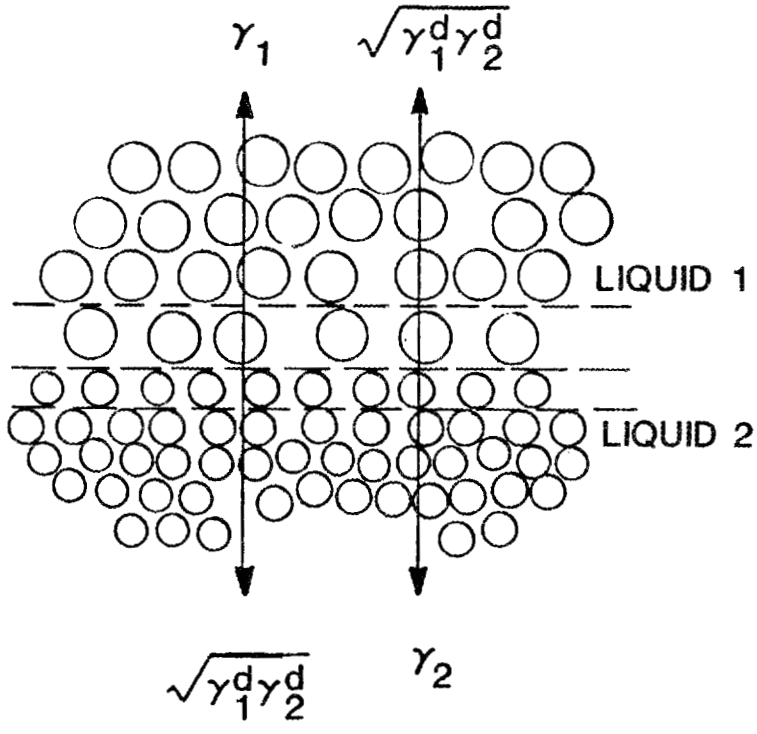

Figure 2. Intermolecular forces acting at fluid-fluid interfaces.

$$|\Delta F_{12}^{a}| = 2\sqrt{\gamma_1 \gamma_2}. \tag{20}$$

Writing the complete energy balance at the interphase in terms of surface and interfacial tension results in the following equation

$$\gamma_{12} = \gamma_1 + \gamma_2 - 2\sqrt{\gamma_1 \gamma_2}. \tag{21}$$

In this equation, γ_1 and γ_2 , as stated by Good and Ebling (1970), represent the surface tension of pure phase components. This equation was found to be a good approximation for fluorocarbon-hydrocarbon systems but not for many water-organic liquid systems. Fowkes (1964) proposed that the surface tension in the square root term should be replaced by the dispersion force-component of the surface tension, γ^d , when dispersion forces predominates in at least one of the liquids and the forces of adhesion are all due to dispersion forces.

It was also recognized by Good and Ebling (1970) that the ratio

$$\Phi = \frac{\Delta F_{12}^{a}}{\Delta F_{12}^{c} \Delta F_{12}^{c}} = \frac{\gamma_1 + \gamma_2 - \gamma_{12}}{2\sqrt{\gamma_1 \gamma_2}} \tag{22}$$

is characteristic of a pair of substances 1 and 2. The generalized equation for interfacial tension is, therefore

$$\gamma_{12} = \gamma_1 + \gamma_2 - 2\Phi\sqrt{\gamma_1 \gamma_2}. \tag{23}$$

Experimental values of Φ were found to range from 0.55 to 1.15 (Good and Ebling 1970). Based on theory (Giriifalco and Good 1957), Φ should be close to unity when the cohesive forces within each liquid and forces acting across the interface are of the same type. When the forces are dissimilar, low values of

Φ are expected. Based on models for intermolecular forces, Good and Ebling (1970) gave an excellent explanation of the theoretical framework for Φ.

Equation 21 was further developed to account for liquids with high mutual solubility (Good and Ebling 1952). The resulting general equation is

$$\gamma_{12} = g_1 + g_2 - 2\Phi_{12}\sqrt{g_1 g_2} \tag{24}$$

where

$g_i = x_{1i}\gamma_1 + x_{2i}\gamma_2,$

$x_{1i} =$ solubility of component 1 in liquid phase i,

$x_{2i} =$ solubility of component 2 in liquid phase i.

Thus, estimations of interfacial tension may be obtained from mutual solubility data, pure component surface tensions, and Φ.

Several empirical expressions have also been suggest for estimating interfacial tension of water-organic systems. Donahue and Bartell (1952) proposed the following formula

$$\gamma_{12} = a - b\ln(x_o^M + x_w^M) \tag{25}$$

vhere

$a = -3.33,$

$b = 7.21,$

x_o^M = mol fraction of organic phase in water, and

x_w^M = mol fraction of water in organic phase.

The correlation is strictly valid only at $25°C$. Donahue and Bartell's correlation has an average error of 15.5% and a maximum error of 48.5%.

Antonov (Gaines 1978) proposed an empirical correlation which uses the surface tension of mutually saturated water and organic phases,

$$\gamma_{12} = |\gamma_{1,2} - \gamma_{2,1}| \tag{26}$$

where

$\gamma_{1,2}$ = surface tension of phase 1 saturated with component 2,

$\gamma_{2,1}$ = surface tension of phase 2 saturated with component 1.

This equation has been criticized by Donahue and Bartell (1952). However, it is generally applicable and reasonably accurate as long as saturated phase surface tensions are used. An average error of 11.8% and a maximum error of 38.6% is obtained from Antonov's rule.

3.4 Thermodynamics of Liquid Interfaces

The basic difficulty in the formulation of the thermodynamics of interfaces is that the interface thickness is unknown (Jaycock and Parfitt 1981). Therefore, an interface must be defined in thermodynamic terms.

Two basic conventions, the surface phase convention and the Gibbs surface convention, are widely used for the thermodynamic treatment of interfaces. The surface phase convention, Fig. 3, assumes that bulk phases α and β are uniform up to the planes AA' and BB', so that all inhomogeneities associated with the interface are restricted to surface phase.

ORNL DWG 87-674

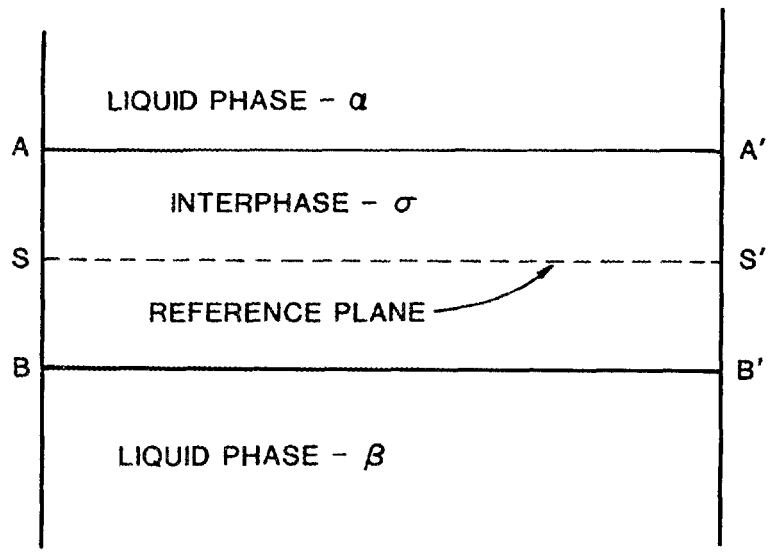

Figure 3. Physical model for thermodynamic surfaces at fluid-fluid interfaces.

Surface concentration, Γ_i^σ of component i, as well as all the extensive thermodynamic quantities, are total amounts associated with σ .

The convention proposed by Gibbs defines the surface as a dividing plane SS', called the Gibbs surface, which is parallel to, and in between, AA' and BB'. Surface excess concentration, i, is the amount of i per unit area in excess of that in unit area in a reference system of the same total volume in which the bulk concentration in α and β remain uniform to the Gibbs surface. The volume of the surface, V^σ, is zero by this convention.

Using the surface phase convention, the following derivation is given for the thermodynamic description of a closed interface (Jaycock and Parfitt 1981). The work done on the interface by the forces across the plane AA' and BB' is $-PAd\tau$. Work done by forces parallel to the planes AA' and BB' is $-(P\tau-\gamma)dA$. The total work done on the interface is

$$\begin{aligned} w_{\text{total}} &= -PAd\tau - (P\tau - \gamma)dA \\ &= -P(Ad\tau + \tau dA) + \gamma dA, \end{aligned} \tag{27}$$

but

$$V^\sigma = \tau A \text{ and}$$

$$dV^\sigma = Ad\tau + \tau dA,$$

therefore

$$w_{total} = -PdV^\sigma + \gamma dA. \tag{28}$$

The first law of thermodynamics states that

$$dU = q + w \tag{29}$$

and making the substitution $q_{\text{reversible}} = TdS$ and putting Eq. 28 into Eq. 29

$$dU = TdS - PdV + \gamma dA. \tag{30}$$

Using the classical thermodynamic definitions,

$$H = U + PV, \tag{30}$$

$$\mathcal{A} = U - TS \tag{32}$$

$$G = U + PV - TS, \tag{33}$$

the following differential equations are obtained for the Helmhotz free energy:

$$d\mathcal{A} = -SdT - PdV + \gamma dA \tag{34}$$

$$\left[\frac{\partial \mathcal{A}}{\partial A}\right]_{T,V} = \gamma, \tag{35}$$

$$\left[\frac{\partial \mathcal{A}}{\partial A}\right]_{V,A} = -S, \tag{36}$$

$$\left[\frac{\partial \mathcal{A}}{\partial V}\right]_{T,A} = -P. \tag{37}$$

At constant temperature

$$d\mathcal{A} = dU - TdS \tag{38}$$

$$= -w_{max} \tag{39}$$

$$= -PdV - \gamma dA \tag{40}$$

if only work associated with volume and area changes is involved.

The differentials for the Gibbs function are:

$$dG = -SdT + VdP + \gamma dA, \tag{41}$$

$$\left[\frac{\partial G}{\partial A}\right]_{T,P} = \gamma, \tag{42}$$

$$\left[\frac{\partial G}{\partial T}\right]_{P,A} = -S, \tag{43}$$

$$\left[\frac{\partial G}{\partial P}\right]_{T,A} = V, \tag{44}$$

and at constant temperature and pressure

$$\begin{aligned} dG &= dU + PdV - TdS, \\ &= -w_{max} + PdV, \\ &\neq \text{ net work.} \end{aligned} \tag{45}$$

For bulk phase systems, it is normal to consider that dG represents the net work of a system. However, as can be seen from Eq. 45, when a system contains an interface, the classical approach does not lead to this equality, since the contribution to the work term of γdA is not in the equation.

This abnormality in the behavior of the Gibbs function led Guggenheim and Hill to redefine H and G (Guggenheim 1967),

$$H = U + PV - \gamma A, \tag{46}$$

$$\begin{aligned} G &= H - TS \\ &= U + PV - \gamma A - TS. \end{aligned} \tag{47}$$

Differentiating Eq. 47,

$$dG = dU + PdV + VdP - \gamma dA - Ad\gamma - TdS - SdT, \tag{48}$$

which for constant T, P and γ yields:

$$\begin{aligned} dG &= dU + PdV - \gamma dA - TdS \\ &= -(w_{max} - PdV + \gamma dA). \end{aligned} \tag{49}$$

The Gibbs function is now equal to net work. For an open system with varying composition, the following classical thermodynamic functions are applicable:

$$dU = TdS - PdV + \gamma dA + \sum_i \mu dn_i, \tag{50}$$

$$dH = dU + PdV + \gamma dA + \sum_i \mu dn_i, \tag{51}$$

$$\begin{aligned} d\mathcal{A} &= dU - TdS - SdT \\ &= -SdT - PdV + \gamma dA + \sum_i \mu_i dn_i, \end{aligned} \tag{52}$$

$$\begin{aligned} dG &= dH - TdS - SdT \\ &= -SdT + VdP + \gamma dA + \sum_i \mu_i dn_i. \end{aligned} \tag{53}$$

Integration of the Gibbs free energy equation at constant intensive variables yields:

$$G = \gamma A + \sum_i \mu_i n_i. \tag{54}$$

This equation is different from the expression for G in the absence of an interface (i.e., bulk phases) in that an extra term γA is included.

Applying the methods of Guggenheim and Hill (1967) to an open system, the following expressions are obtained by redefining G and H:

$$dH = TdS + VdP - Ad\gamma + \sum_i \mu_i dn_i, \text{and} \tag{55}$$

$$dG = -SdT + VdP - Ad\gamma + \sum_i \mu_i dn_i. \tag{56}$$

The expressions for $d\mathcal{A}$ and dU are the same as those derived from the classical method. Integration of Eq. 56 at constant intensive variables yields:

$$G = \sum_i \mu_i n_i \quad \text{and} \tag{57a}$$

$$dG = \sum_i \mu_i dn_i + \sum_i n_i d\mu_i, \tag{57b}$$

which is of the same form as that for bulk phases.

The redefinition of H and G by Guggenheim and Hill lead to a more consistent treatment of interphases than may be obtained by classical methods. Guggenheim and Hill's approach results in thermodynamic parameters which are not altered by the presence of an interface. Therefore, their method should be viewed as precise generalized definitions, while the classic treatment should be viewed as specific cases for bulk phases in the absence of an interface.

Using the Guggenheim-Hill definitions for dG (Eqs. 55 and 57b) and equating the two, the following equation may be obtained:

$$SdT - VdP + Ad\gamma + \sum_i n_i d\mu_i = 0 \tag{58}$$

This equation is analogous to the Gibbs-Duhem equation for a bulk phase which requires no $Ad\gamma$ term.

The Gibbs adsorption isotherm which relates the extent of adsorption of components at the interface to the interfacial tension is easily derived from Eq. 58. At constant temperature and pressure, Eq. 58 reduces to

$$-Ad\gamma = \sum_i n_i d\mu_i, \tag{59}$$

where

n_i = mol of component i in the interfacial region.

The parameter Γ_i ,the amount of species i in a unit area of the interface, is expressed mathematically:

$$\Gamma_i = \frac{n_i}{A}. \tag{60}$$

Substitution of Eq. 60 into Eq. 59 yields the familiar Gibbs adsorption isotherm equation:

$$-d\gamma = \sum_i \Gamma_i d\mu_i. \tag{61}$$

The Gibbs-Duhem equations for the adjoining phases (α and β) are:

$$\sum_i n_i^\alpha d\mu_i^\alpha = 0, \tag{62}$$

$$\sum_i n_i^\beta d\mu_i^\beta = 0 \tag{63}$$

Adopting the surface phase convention and applying the Gibbs-Duhem equations (Eqs. 62 and 63) for bulk phases, the following equation applies to

a two phase system (α and β) with i components and with 1 and 2 being the "solvents" in α and β, respectively:

$$-d\gamma = \sum_{j=3}^{i} \left[(\Gamma_2^s - \frac{x_2^\alpha}{x_1^\alpha}\Gamma) \left(\frac{x_j^\alpha x_1^\beta - x_j^\beta x_1^\alpha}{x_1^\alpha x_2^\beta - x_1^\beta x_2^\alpha} \right) + (\Gamma_j^s - \frac{x_j^\alpha}{x_1^\alpha}\Gamma_1^s) \right] d\mu_j \qquad (64)$$

where the x are mol fractions. For two immiscible solvents (i.e. $x_1^\beta = x_2^\alpha = 0$), with a surface-active solute 3 which is soluble in phase α only (i.e. $x_3^\beta = 0$), Eq. 64 reduces to

$$-d\gamma = (\Gamma_3^s - \frac{x_3^\alpha}{x_1^\alpha}\Gamma_1^s) d\mu_3 . \qquad (65)$$

In the case of dilute solutions and if Γ_1^s is negligible, Eq. 65 becomes

$$-d\gamma \approx \Gamma_3^s d\mu_3 . \qquad (66)$$

Guggenheim (1967) warns that the term $\frac{x_3^\alpha}{x_1^\alpha}\Gamma_1^s$ in Eq. 65 should not be omitted even in the case of a dilute solution, because although $x_3^\alpha \ll x_1^\alpha$, it is possible for $\Gamma_1^s \gg \Gamma_3^s$. A similar equation to Eq. 66 is obtained if the surface-active solute 3 is soluble in both phases and both phases contain dilute solutions of component 3.

In the special case where α is a liquid phase aqueous solution and air is the vapor phase, β , the surface tension as a function of the solute concentration in the liquid phase exhibits the three basic forms shown is Fig. 4. Curves 1 and 3 indicate positive adsorption of the solute, i.e. Γ_3^σ, surface excess (Gibbs surface convention) is positive. Curve 2 indicates negative surface excess, or a surface depletion of the solute (Hiemenz 1986).

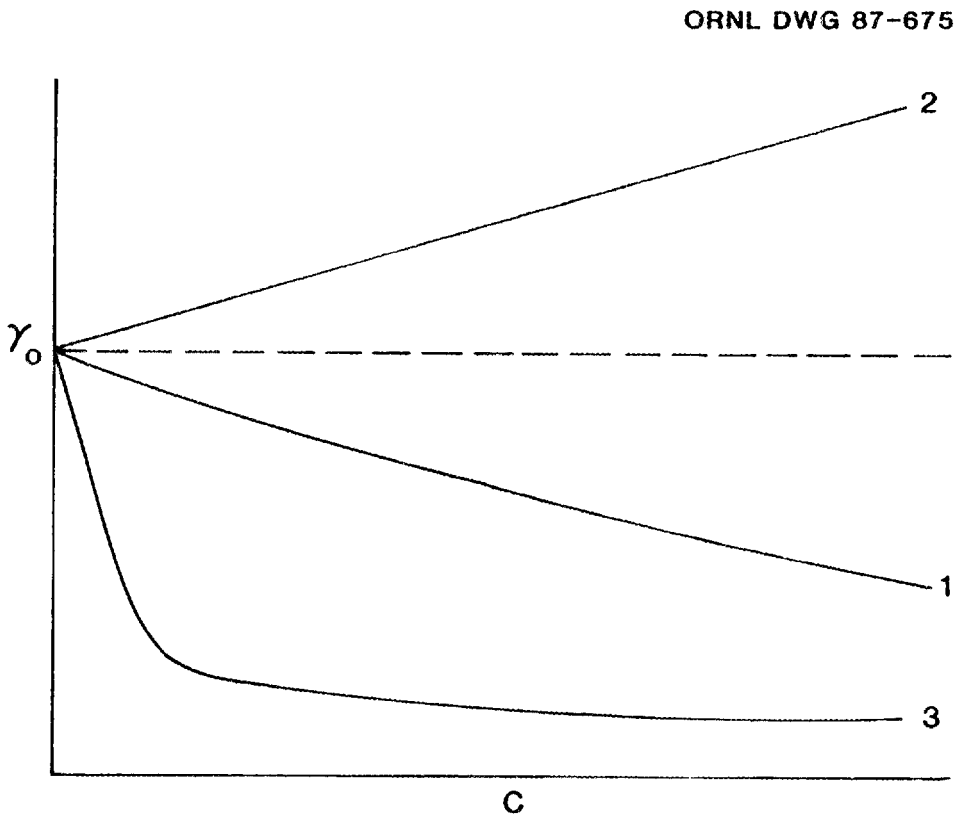

Fig. 4. Effect of surfactant on surface tension: (1) simple organic solutes, (2) simple electrolytes, (3) amphipathic solutes.

Curve 1 is typical of the effect that most unionized organics have on surface tension. Curve 2 is typical of aqueous solutions containing inorganic electrolytes and highly hydrated organic compounds. The type of behavior shown in curve 3 is characteristic of soluble amphipathic species. It is also observed that interfacial tension between two liquids behaves in a similar manner.

The temperature dependence of interfacial tension for a two component system can be easily derived from Eq.58 and applying the Gibbs-Duhem equation for each of the liquid phases. Neglecting the term VdP, the following Gibbs-Duhem relation in each of the liquid phases is given:

$$\overline{S}_A^{\alpha} dT + (1 - x^{\alpha}) d\overline{\mu}_1 + x d\overline{\mu}_2 = 0 \quad \text{and} \tag{67}$$

$$\overline{S}_A^{\beta} dT + (1 - x^{\beta}) d\overline{\mu}_1 + x^{\beta} \overline{\mu}_2 = 0. \tag{68}$$

To obtain the temperature dependence of γ, $d\overline{\mu}_1$ and $d\overline{\mu}_2$ are eliminated from Eqs. 58, 67 and 68 yielding the following equation:

$$\frac{d\gamma}{dT} = \frac{\begin{vmatrix} \overline{S}_A^{\sigma} & \overline{S}_A^{\alpha} & \overline{S}_A^{\beta} \\ \Gamma_1 & 1 - x^{\alpha} & 1 - x^{\beta} \\ \Gamma_2 & x^{\alpha} & x^{\beta} \end{vmatrix}}{x^{\beta} - x^{\alpha}} \quad \text{or}$$

$$\frac{d\gamma}{dT} = \frac{\begin{vmatrix} \overline{S}_A^{\sigma} & \overline{S}_A^{\alpha} & \overline{S}_A^{\beta} \\ \Gamma_1 + \Gamma_2 & 1 & 1 \\ \Gamma_2 & x^{\alpha} & x^{\beta} \end{vmatrix}}{x^{\beta} - x^{\alpha}} \tag{69}$$

For the special case where the two liquids are immiscible, the temperature coefficient of γ is related to the specific interfacial entropy, $\overline{S}$, and internal energy, $\overline{U}$, by the equations

$$\frac{d\gamma}{dT} = \overline{S}^{\sigma} - \Gamma_1 \overline{S}^{\alpha} - \Gamma_2 \overline{S}^{\beta} = \overline{S}_{\gamma}^{\alpha\beta} \text{ and} \tag{70}$$

$$\gamma - T\frac{d\gamma}{dT} = \overline{U}^{\sigma} - \Gamma_1 \overline{U}^{\alpha} - \Gamma_2 \overline{U}^{\beta} = \overline{U}_{\gamma}^{\alpha\beta} \tag{71}$$

where

$S_{\gamma}^{\alpha\beta}$ = the entropy of interface formation and

$U_{\gamma}^{\alpha\beta}$ = the internal energy of interface formation.

3.5 The Pendant Drop Method

The shape of a pendant drop is governed by the Laplace equation of capillarity

$$\Delta P = \gamma(\frac{1}{R_1} + \frac{1}{R_2}). \tag{72}$$

At the vertex of the drop, the radii of curvature are equal, and the equation becomes $R_1 = R_2 = R_o$ at $x = y = 0$ (where x and y are the horizontal and vertical coordinates of any point in a meridional section of the surface of the fluid, and R_o is the radius of curvature at the vertex. It also follows that, the pressure at $y = 0$ is given by $\Delta P = \frac{2\gamma}{R_o}$. But at other values of y, ΔP due to hydrostatic pressure is equal to $\Delta\rho g y$; where $\Delta\rho$ is the density difference across the two interfaces, and g is the gravitational constant. The resulting equation is:

$$\gamma\left(\frac{1}{R_1} + \frac{1}{R_2}\right) = -\Delta\rho g y + \frac{2\gamma}{R_o} \tag{73}$$

It is common to nondimensionalize Eq. 72, by introducing the parameter

$$\beta = -\frac{\Delta\rho g R_o^2}{\gamma} \tag{74}$$

and substituting $Y = y/R_o$, and β into Eq. 73 to get

$$R_o\left(\frac{1}{R_1} + \frac{1}{R_2}\right) = \beta Y + 2. \tag{75}$$

Noting the geometrical meaning of R_1, and R_2, in Fig. 5, it can be shown that Eq. 75 can be expressed by the following equation in X and Y, where $X =$ dimensionless horizontal coordinate, x/R_o :

$$\frac{Y''}{(1+Y'^2)^{3/2}} + \frac{Y'}{X(1+Y'^2)^{1/2}} = \beta Y + 2. \tag{76}$$

This equation is a nonlinear second order ordinary differential equation which must be solved numerically. The boundary conditions are at $X = 0, Y = 0$, and $Y' = 0$.

Bashforth and Adams (1883) suggested resolution of the equation by incremental series around the surface of the drop. However, because of the slope-dependent variable associated with using numerical methods on Eq. 76, another form of Eq. 76 that uses arc length, s, as the independent variable was suggested by Bashforth and Adams.

Starting with Eq.73 and the definitions of R_1 and R_2 (Fig. 6), the transformation of Eq. 73 into an arc-length-dependent equation is readily obtained. Since $R_2 = \frac{x}{\sin\phi}$, Eq. 72 becomes

$$\gamma\left(\frac{1}{R_1} + \frac{\sin\phi}{x}\right) = \Delta\rho g y + \frac{2\gamma}{R_o}, \tag{77}$$

ORNL DWG 87-676

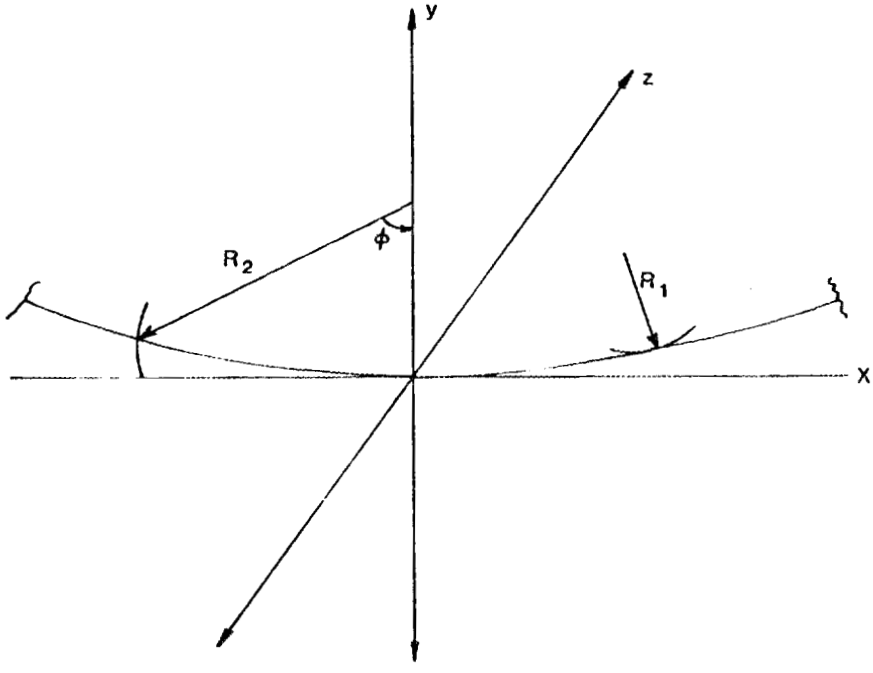

Fig. 5. Graphic representation of principal radii.

ORNL DWG 87-678

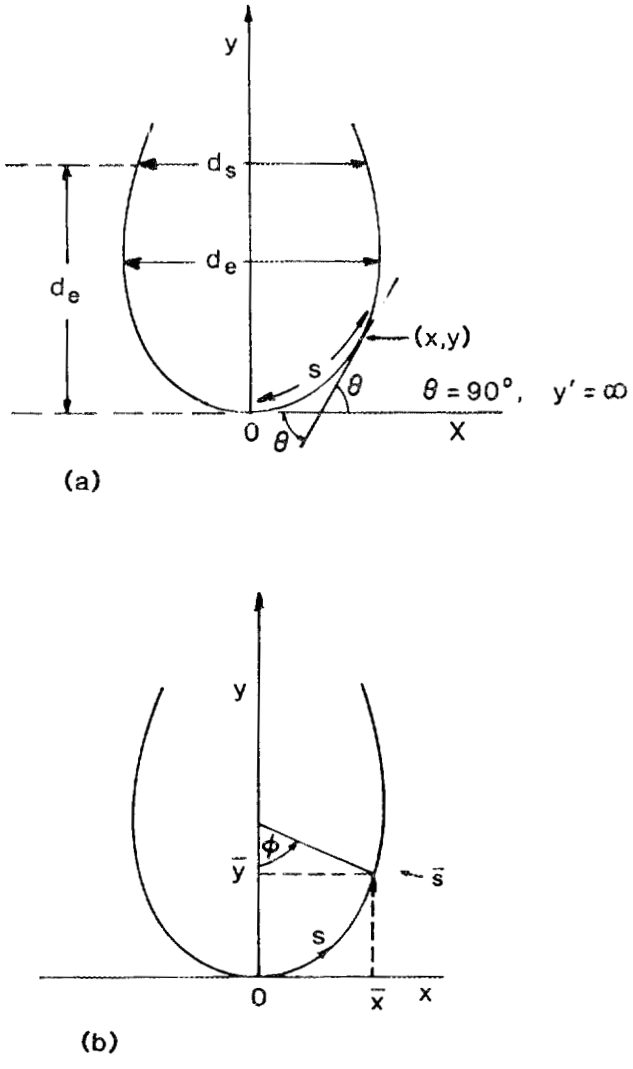

Fig. 6. Pendant Drop: (a) Definition of D_c and D_s, (b) alternative coordinate systems.

and the governing equations for using s as the independent variable are:

$$x' = \frac{dx}{ds} = \cos\phi \text{ and} \tag{78}$$

$$y' = \frac{dy}{ds} = \sin\phi \tag{79}$$

and by definition,

$$\frac{1}{R_1} = \frac{d\phi}{ds}. \tag{80}$$

Substituting Eq. 80 into Eq.77 , the resulting equation is

$$\phi' = \frac{d\phi}{ds} = \frac{2}{R_o} + \frac{\Delta\rho g y}{\gamma} - \frac{\sin\phi}{x} \tag{81}$$

or

$$\phi' = \frac{d\phi}{ds} = \frac{2}{R_o} + \beta_c y - \frac{\sin\phi}{x} \tag{82}$$

where $\beta_c = \frac{\Delta\rho g}{\gamma}$. Equations 78, 79, and 81 form a set of first-order differential equations with x, y, and ϕ as functions of the independent variable s. The boundary conditions are

$$x(0) = y(0) = \phi(0) = 0. \tag{83}$$

Eq. 81 is usually expressed in the following dimensionless form:

$$\phi' = \frac{d\phi}{dS} = 2 + \frac{\Delta\rho g R_o^2 Y}{\gamma} - \frac{\sin\phi}{X} \tag{84}$$

where,

$$X = \frac{x}{R_o},$$
$$Y = \frac{y}{R_o},$$
$$S = \frac{s}{R_o}.$$

Substituting the shape parameter, β, as is defined by Eq. 76, this equation becomes

$$\phi' = 2 + \beta Y - \frac{\sin\phi}{X}. \tag{85}$$

The shape parameter, β, cannot be measured directly and conveniently with any great accuracy; but as a shape determining parameter, it can be related to other variables that can be determined more easily and more accurately.

Andreas, Hauser and Tucker (1938) suggested measuring the drop at two selected planes, D_e, the diameter at the equatorial plane, and D_s, the diameter at the plane D_e from the vertex of the drop. The ratio, $\frac{D_s}{D_e}$ or $\mathcal{S}$, is a function of β. Furthermore, Andreas et al. also defined the following new quantity, $\mathcal{H}$, to remove the parameter R_o from Eq. 75:

$$\mathcal{H} = -\beta \left(\frac{D_e}{R_o}\right)^2 \tag{86}$$

Substitution Eq. 74 into Eq. 86 and solving for γ, the following equation is obtained:

$$\gamma = \frac{\Delta\rho g D_e^2}{\mathcal{H}}. \tag{87}$$

The common method for determining β is to determine $\mathcal{S}$ from measurements of D_s and D_e and to obtain $\mathcal{H}$ from tables of $\frac{1}{\mathcal{H}}$ vs. $\mathcal{S}$. Fordham (1948), and Niederhauser and Bartell (1950) independently solved the fundamental equation (Eq. 74) in order to obtain a theoretically sound table of $\frac{1}{\mathcal{H}}$ vs. $\mathcal{S}$. Their numerical integration techniques were similar to the method of Bashforth and Adams (1883). This method of determining γ is simple, but it requires precise measurements of D_e and D_s. The precision of this method as published by Andreas,

Hauser, and Tucker (1938) is approximately one percent.

Recent techniques for determining interfacial tension by the pendant drop method entail correlations of the entire data on the drop shape to the theoretical Laplace curve. These techniques should result in better estimates of γ, since more data points are used to obtain this parameter. The most common methods use numerical techniques to simultaneously integrate the three ordinary differentials of Bashforth and Adams(Eqs. 77, 78 and 80). The best combination of R_o and β_c is found by nonlinear parameter estimation techniques such that the defined objective function is minimized. Maze and Burnet (1969 and 1971) were the first to develop a satisfactory scheme for the determination of interfacial tension from the drop shape coordinates. The objective function was to minimize the horizontal distances between the measured and calculated curve. This type of objective function can, however, lead to serious error if the shape tends to flatten near the apex. Any data point which is near the apex, may cause a large error even if it lies very close to the "best" curve; thus, leading to bias of the solution. Rotenberg, Boruvka, and Neumann (1981) have, therefore, defined the objective function, E, as the sum of the perpendicular distances between the experimental data points and the theoretical curve. In theory, there should be only one set of unbias parameters, β_c and R_o, where the sum of the perpendicular distances is minimized. The algorithm by Rotenberg, Boruvka and Neumann (1981) also takes into account the importance of locating the apex of the drop, which determines the origin of the calculated curve. Their model consists, therefore, of four parameters X_o, Y_o, R_o, and β_c, which are varied to minimize the objective function. Numerical integration is done by a Second-Order Implicit Euler Method, and changes in parameter estimates are

obtained by the Newton-Raphson method with incremental loading of one of the parameters. Analytical expressions are used for the derivatives of the objective with respect to each parameter and parameter combinations.

The theoretical aspects of the nonlinear parameter estimation method that is used in the present study is given in the following section. This method uses the same objective function that is given by Rotenberg (1981); however, the method differs in the method of performing numerical integration and in the use of numerical derivatives rather than analytical derivatives. Futhermore, good initial parameter estimates of β_c, R_o, and the coordinates of the origin (i.e. X_o and Y_o) are obtained during the digitization step. Therefore, the present algorithm converges in 5 to 10 iterations. Algorithms, such as Rotenberg's, may require a large number of iterations ($\approx$50) to converge. The present algorithm is also written generically, such that the effect of external fields (in addition to gravity) on the pendant drop may be incorporated into the model.

3.6 Nonlinear Parameter Estimation

There are a number of different methods for obtaining nonlinear parameter estimates. The three most widely used computer based routines include (1) linearization, (2) steepest decent, and (3) Marquardt's compromise (Draper and Smith 1981). Of these three, the linearization (or Taylor series) method is employed in this study because of its general use to successfully solve many nonlinear problems.

The linearization method uses the results of linear least squares in a succession of steps. This is shown by assuming that the proposed model is given

by Eq. 88

$$Y^E = f(\xi_1, \xi_2, ..., \xi_k; \theta_1, \theta_2, ...\theta_p) + \epsilon, \tag{88}$$

where

$$\xi_k = \text{ the independent variable,}$$

$$\theta_p = \text{parameter estimates,}$$

and choosing the initial parameter estimates, θ_1°, θ_2°, ..., θ_n° for the parameters $\theta_1, \theta_2, ..., \theta_n$. A Taylor series expansion of $f(\xi_j, \theta)$ is carried out about the point θ° (i.e., θ_1°, θ_2°,.....,θ_p°). The series is then truncated after the first derivatives,

$$f(\xi_j, \theta) = f(\xi_j, \theta^\circ) + \sum_{i=1}^{p} \left[\frac{\partial f(\xi_j, \theta)}{\partial \theta_i} \right]_{\theta = \theta_\circ} (\theta_i - \theta_i^\circ) \tag{89}$$

where

$$\xi = \begin{vmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1j} \\ \xi_{21} & \xi_{22} & \cdots & \xi_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{n,1} & \xi_{n,2} & \cdots & \xi_{n,j} \end{vmatrix} \tag{90}$$

Letting

$$f_j^\circ = f(\xi_j, \theta^\circ),$$

$$\Delta_i^\circ = \theta_i - \theta_i^\circ,$$

$$Z_i = \left[\frac{\partial f(\xi_j, \theta)}{\partial \theta_i} \right]_{\theta = \theta^\circ},$$

Eq. 89 becomes

$$Y^E - f_j^\circ = \sum_i \Delta_i Z_{ij}^\circ + \epsilon_j. \tag{91}$$

Eq. 91 is linear in terms of θ and therefore the linear least squares theory may be applied. In vector notation, $\mathbf{b}_\circ^p$, the estimates of Δ° are obtained by solving the following equation:

$$\mathbf{b}_\circ^p = (\mathbf{Z}'^\circ \mathbf{Z}^\circ)^{-1} \mathbf{Z}'^\circ (\mathbf{Y}^E - \mathbf{f}^\circ). \tag{92}$$

Since linear least squares theory is applied, $\mathbf{b}_\circ^p$, will minimize the sum of squares

$$SS(\theta) = \sum_{j=1}^{n} \left[Y_j^E - f(\xi_j, \theta^\circ) - \sum_{i=1}^{p} \Delta_i^\circ Z_{ij}^\circ \right]^2 \tag{93}$$

with respect to Δ_i°, $i = 1, 2, \ldots, p$. New values of $\theta_{i(k+1)}$ are obtained by letting $\theta_{k+1} = \theta_k + \mathbf{b}_k$, where k is the number of iterations. This iterative process is continued until the solution converges, that is, until in successive iterations k, (k +1),

$$\left| \frac{\theta_{i(k+1)} - \theta_{ik}}{\theta_{ik}} \right| < \delta \tag{94}$$

or

$$\left| \frac{SS(\theta_{k+1}) - SS(\theta_k)}{SS(\theta_k)} \right| < \delta_s, \tag{95}$$

where δ and δ_s are some specified small amount (e.g., 0.000001). At each stage of the iterative process, $SS(\theta)$ should be evaluated to assess whether a reduction in its value has actually been achieved.

The linearization method has several possible drawbacks for some problems in that:

1. It may converge very slowly, i.e., a very large number of iterations may be required before the solution stabilizes even though the sum of squares $SS(\theta)$ may decrease consistently as k increases. This sort of behavior is not common but can occur.

2. The estimate may oscillate widely, continually reversing direction and often increasing, as well as decreasing the sum of squares. Nevertheless, the solution may eventually stabilize.

3. It may not converge at all, and even diverge, so that the sum of squares increases without bound. This will probably occur if the initial estimates are not good guesses or if the proposed model is incorrect.

As is seen from the definition of **Z**, it is necessary to compute the partial derivative of the function f with respect to each of the parameters. This may be done by determining the analytical expression for each of the derivatives which, in some cases may be difficult to obtain, or by computing the ratio

$$\frac{f(\xi_j;\theta_1^\circ,\theta_2^\circ,...,\theta_{i+h_i}^\circ,...,\theta_p^\circ)-f(\xi_j;\theta_1^\circ,\theta_2^\circ,...,\theta_i^\circ,...,\theta_p^\circ)}{h_i} \tag{96}$$

where $i = 1, 2, ...p$, and h_i is a small increment. A number of nonlinear computer algorithms employ the latter method to compute the derivatives.

At this point, it is helpful to compare the sum of squares surfaces for linear and nonlinear models. Figs. 7a and 7b depicts the sum of squares surfaces for linear and nonlinear models, respectively. The surface contour for the linear is ellipsoidal and has a single local and global minimum height, $S(\theta)$, at the least squares estimator, $\hat{\theta}$. If the model is nonlinear, the contours are not ellipsoidal but tend to be irregular and often "banana-shaped." The model may also consists

of several local minima and perhaps with more than one global minimum.

The linearization technique converts the problem of finding the minimum height of $S(\theta)$ for a nonlinear model starting from an initial point θ°, into a series of linear models. The initial linearization of $f(\xi, \theta)$ about θ° replaces the irregular $S(\theta)$ bowl by an elliptical bowl $SS(\theta)$ with a "psuedo right shape," i.e., it has the same first derivatives of the corresponding model function right at θ°. The accuracy with which this technique approximates the actual $S(\theta)$ contours depends on (1) the model assumed, (2) the data available, and (3) the relative positions of θ° and $\hat{\theta}$ in the θ-space. In any event, the linearization method solves the problem at θ° by moving to the bottom point of the linearized bowl at θ° to reach θ as shown in Fig. 8. The process is repeated at θ_1. With successive iterations, it is hopeful that the procedure will converge to $\hat{\theta}$, as is shown in Fig. 9.

ORNL DWG 87-686

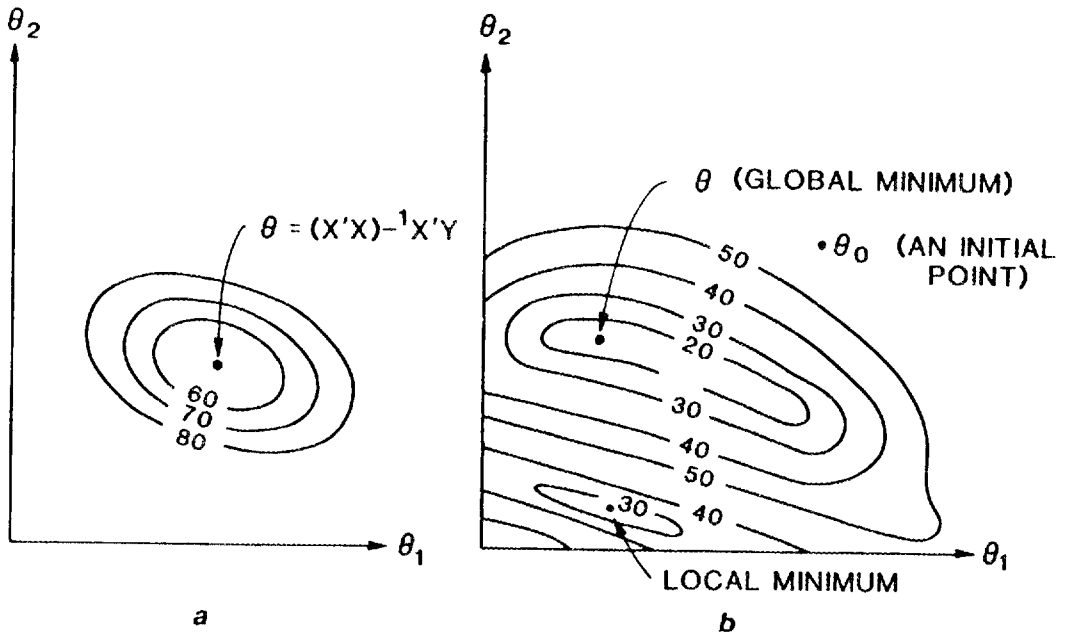

Fig. 7. Sum of squares surfaces for linear and nonlinear models: (a) linear model (b) nonlinear model.

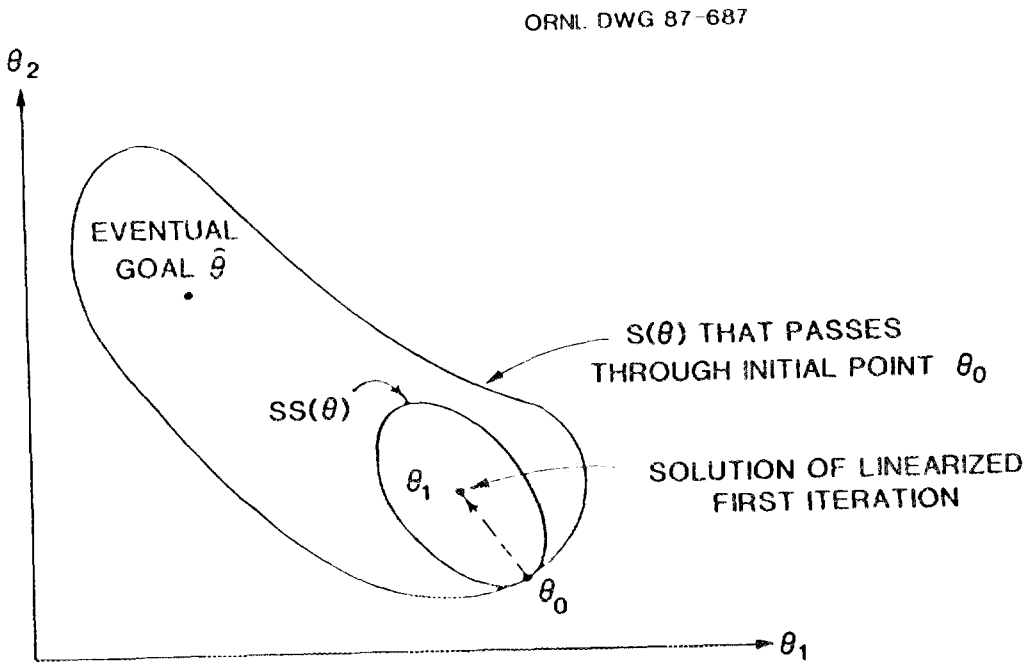

Fig. 8. Comparison between nonlinear and linearized first iteration.

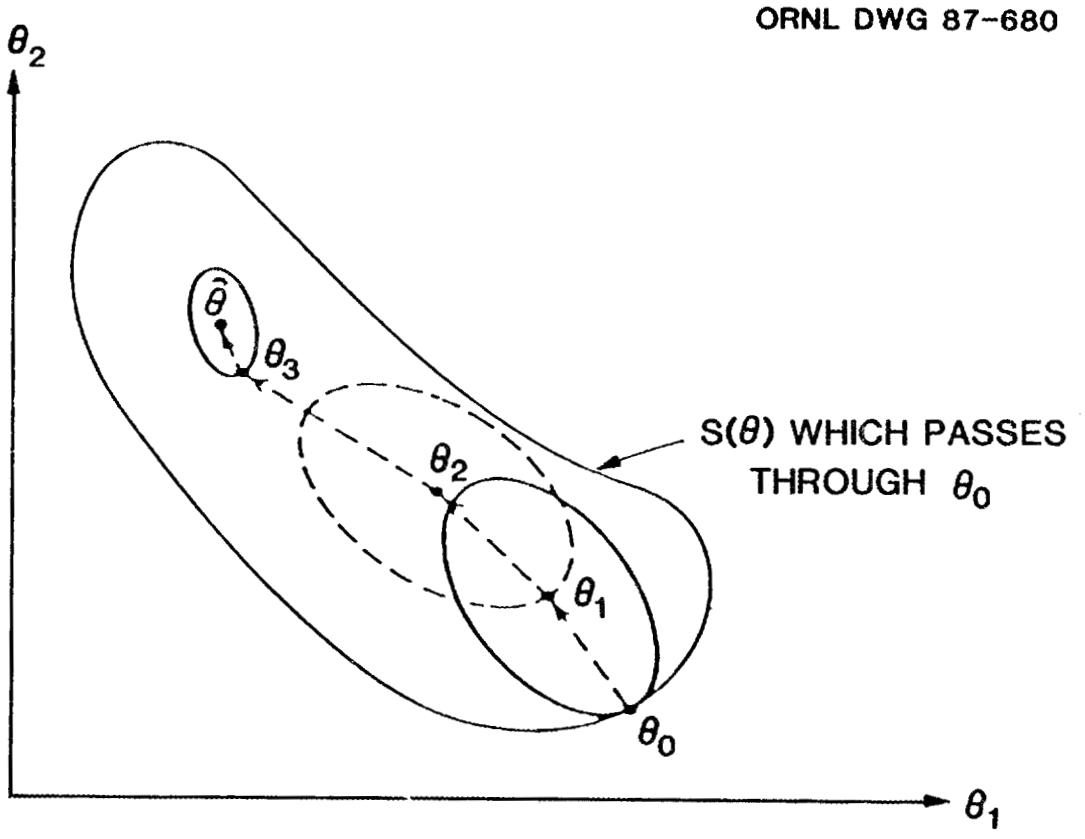

Fig. 9. Successive iteration of linearized model.

3.7 DETERMINING INTERFACIAL TENSION BY NONLINEAR PARAMETER ESTIMATION

As mentioned above, the convergence of a nonlinear parameter estimation (NLPE) algorithm to the global minimum depends upon the model assumed, the data available, and the relative positions of θ° and $\hat{\theta}$ in the θ-space. Therefore, it is necessary to define a model that adequately describes the data and allows convergence to the global minimum. Rotenberg defined the objective function as the square of the perpendicular distance between the experimental data and the theoretical Laplace curve. The objective function is expressed mathematically as

$$f(\xi_i, \theta) = \left[(X_{obs,i} - X_{calc,i})^2 + (Y_{obs,i} - Y_{calc,i})^2\right]. \tag{97}$$

Since the objective function contains the measured (observed) values for variables X and Y, both variables are defined as independent variables. A dependent variable must therefore be indentified in order to perform nonlinear parameter estimation. The observed perpendicular distances between the measured (digitized) data points and drop surface coordinates are zero. That is, the measured dependent variable, d_k^2, equals zero.

The values X_{calc} and Y_{calc} are determined by simultaneous numerical integration of the Bashforth-Adams equations and testing for perpendicularity. Numerical integration is done by the Fourth-Order Runge-Kutta method for simultaneous first ordinary differential equations (Scarborough 1966).

The method of testing for perpendicularity is shown in Fig. 10.
As is shown, if the sign of the slope of the line

ORNL DWG 87-688R

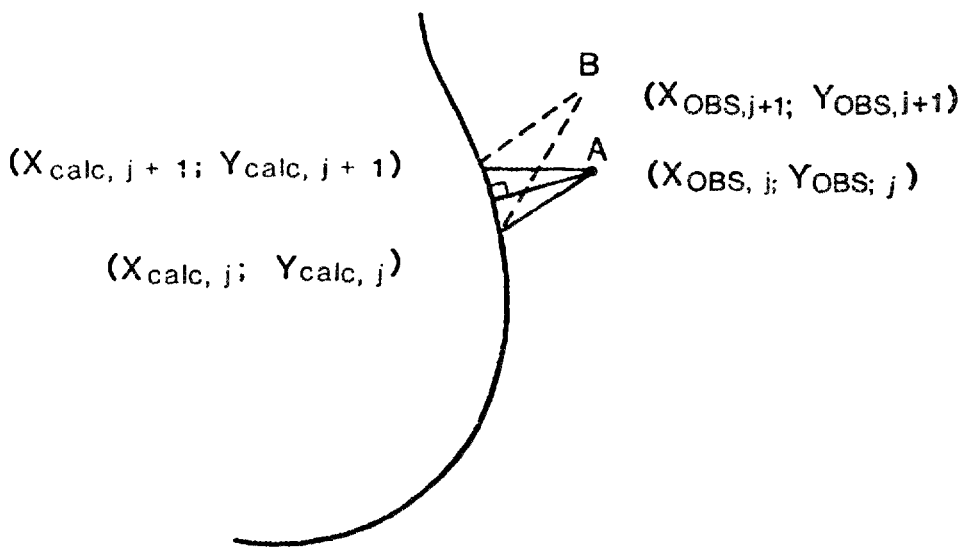

Fig. 10. Test for perpendicularity.

$$[(X_{calc,j}, Y_{calc,j}); (X_{obs,i}, X_{obs,i})]$$

and the sign of the slope of the line

$$[(X_{calc,j+1}, Y_{calc,j+1}); (X_{obs,i}, Y_{obs,i})]$$

are not the same, then there is a point between j and $j+1$ calculated values where the experimental data is perpendicular to the theoretical curve. However, if the sign of the slopes are the same, there is not a point between the j and j+1 calculated data point where the experimental data point is perpendicular to the theoretical curve. An interpolation technique is then used to determine the calculated X and Y data point that is perpendicular to the experimental datum (Bridger 1977). Finding the perpendicular distance between the experimental and theoretical curve is the most computer– time–intensive step.

To complete the information required for nonlinear parameter estimation, it is necessary to compute the derivatives of $f(\xi, \theta)$ with respect to each of the parameters (β_c, R_o, X_o, and Y_o). The programs NONLI4.BAS and PENDR.F (Appendix C) computes these derivatives by Eq. 96. The value of h_i is $0.00005 \times \theta_i$. Analytical expressions are used to compute derivatives in the program NONLI8.BAS. The equations are the same as those given by Rotenberg et. al (1981).

With the above information, $\mathbf{b}_o^p$, is computed by matrix manipulation of the vector representation. The new values in the parameter vector, θ, are computed by the equation

$$\theta = \theta^\circ - f_r \mathbf{b}_o^p, \tag{98}$$

where f_r represents the fraction of $\mathbf{b}_o^p$ to be used to compute the new value of θ. In most cases, f_r can be set at one; however, in cases where the initial parameter estimates are not close to the solution, or if the algorithm diverges, it is sometimes necessary to select f_r to be less than one.

The above procedure for computing interfacial tension has been incorporated into a modified version of a public domain algorithm for nonlinear parameter estimation. The original program was written by Whitman (1982) in Microsoft BASIC for the IBM PC and is based on a paper by Wentworth (1965). The program, NONLI4.BAS, has been modified, translated into the FORTRAN 77 language, and ported to a mainframe computer.

4. EXPERIMENTAL

4.1 Pendant Drop Apparatus and Drop Formation

Initial experiments with water-air, water-cyclohexane, and water-octanol systems were conducted in a system described by Malinauskus and Byers (1985). Droplets were suspended from 0.95 cm (0.375 in.)-Teflon, aluminum-coated nozzles with a single bore at its tip. After several uses, the droplets began to wet the surface of the tip. This resulted in a (r/a_{cc}) ratio that was greater than the recommended value of 0.5 or less. The term r is the radius of the nozzle if the liquid wets the nozzle, and r is the radius of the bore if the liquid does not wet the nozzle (Jennings 1957). The parameter a_{cc} is the capillary constant given by

$$a_{cc} = \sqrt{\frac{2\gamma}{\Delta\rho g}}. \tag{99}$$

Furthermore, the above setup required large volumes of liquids (600 mL) for

the continuous phase which made temperature control to within $\pm 0.1^\circ C$ very difficult.

Another column was, therefore, designed which required a smaller quantity of continuous phase liquid (50 mL) and allowed better temperature control. The pendant drop column consists of 1.27 cm (0.5 in.) square glass tubing, jacketed by 2.54 cm (1.0 in.) square tubing (Fig. 11).

This arrangement allows for a temperature control of the system to within $\pm 0.1^\circ C$ by circulating ethylene glycol through the annulus. The temperature of the ethylene glycol was controlled by flow through a Haake constant temperature bath. A DORIC Trendicator (400A Type K/$^\circ C$) temperature recorder with chromel-alumel thermocouples was used to monitor the temperature.

Droplets were dispensed by a chemically inert microsyringe (Cole-Palmer J-7844-00, 2.0 mL max. volume) and were suspended from a 0.125 cm OD stainless steel syringe needle. The microsyringe was filled by immersing the needle into the bottom phase. Air bubbles were removed by inverting the syringe and displacing any gas. The syringe was quickly transferred to the pendant drop column where stable drops were formed.

4.2 Video Recording and Digitization

A high speed video camera (Tritronics model PC5600) was employed to capture the droplet image. Shutter speeds could vary from 4 to 0.1 ms with framing rates of 60 to 300 frames per second. The camera was equipped with an easily focused telephoto lens (Micro-Nikkor 105mm f/2.8) for closeup photography. The camera was interfaced with a Sony VO-5800 video recorder, which

ORNL-Photo 2437-87

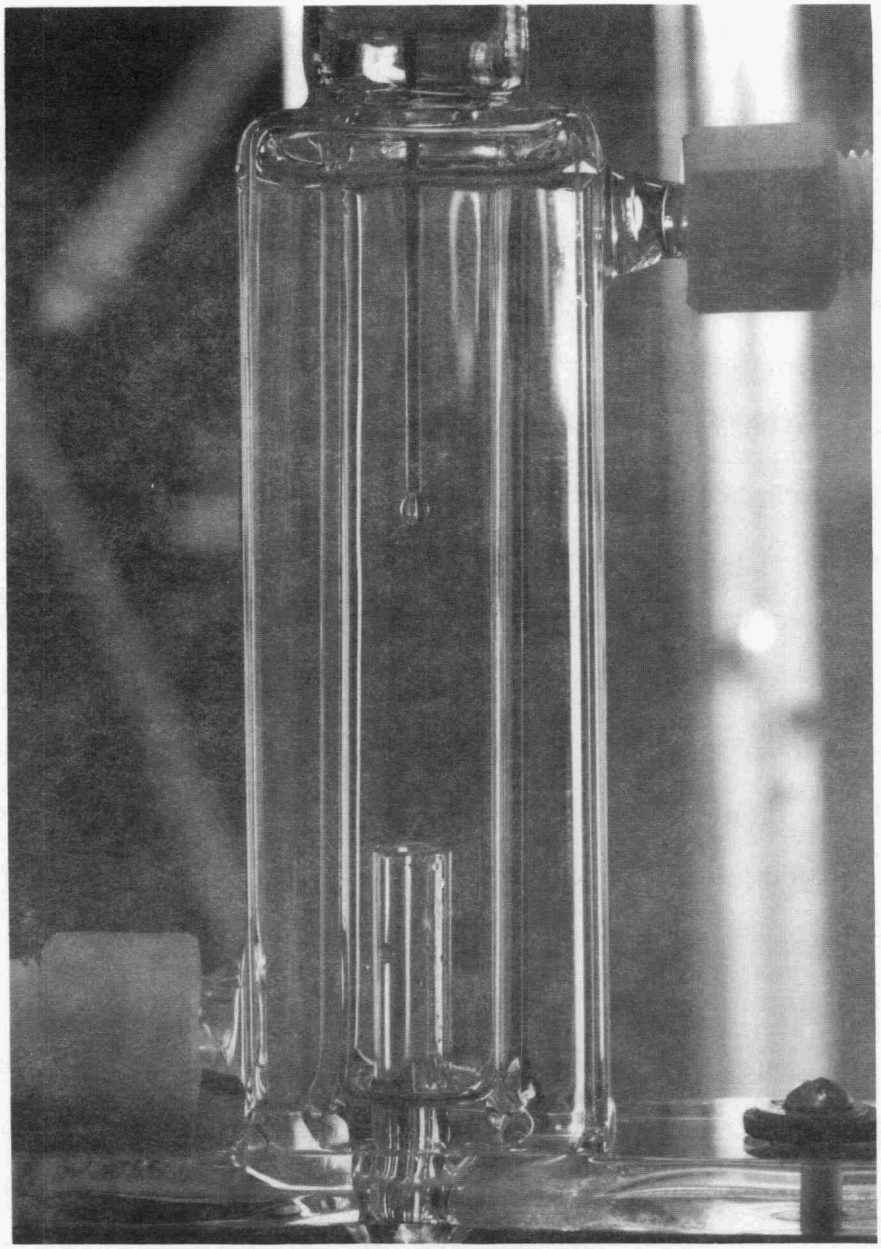

Fig. 11. Photograph of pendant drop column

was used for recording the video image and playback of the image for analysis (Figs. 12 and 13).

After a drop was suspended for three minutes, images of the droplets were tape recorded for 30 to 60 s. The position of the image on the tape was defined by the counter reading on the recorder. Horizonal and vertical scaling factors were determined by suspending into the plane of view either a 0.3175 cm (0.125 in.) spherical rotameter ball or a "T" consisting of two perpendicular sections of 0.1588 cm (0.0625 in.) stainless steel rods.

Backlighting of the image was arranged to give the best detection of the droplet profile. A Lowel Softlight 2 (2000 W maximum power) was the lighting source. Light control was obtained by setting the light intensity via powerstat control and adjusting the "barn doors" for flare control.

Digitization of the drop profile was done by a manual trace with a Sony SMI-70 Graphics computer with Genlocking capabilities (Fig. 13). The Genlocker allowed microcomputer graphics to overlay a video image. Sony Graphics Editor, a software package for the SMI-70, utilized the Genlocker capabilities for manual tracing of a drop profile with commands from the keyboard. To transfer these graphics data into a file of x and y coordinates, a procedure was developed to trace the perimeter of the drop in one color, and subsequently, to fill the interior of the drop with a different color. Thus, a computer graphics picture was generated and stored on 3.5 in. disks. A basic program for the SMC–70, PFIND4, was then used to obtain x,y coordinates for the drop profile by loading the graphics picture and scanning the entire graphics screen to determine the drop edge. This procedure took about 4 to 5 min to complete.

The digitized data was transferred from the Sony graphics computer to an

ORNL-Photo 2435-87

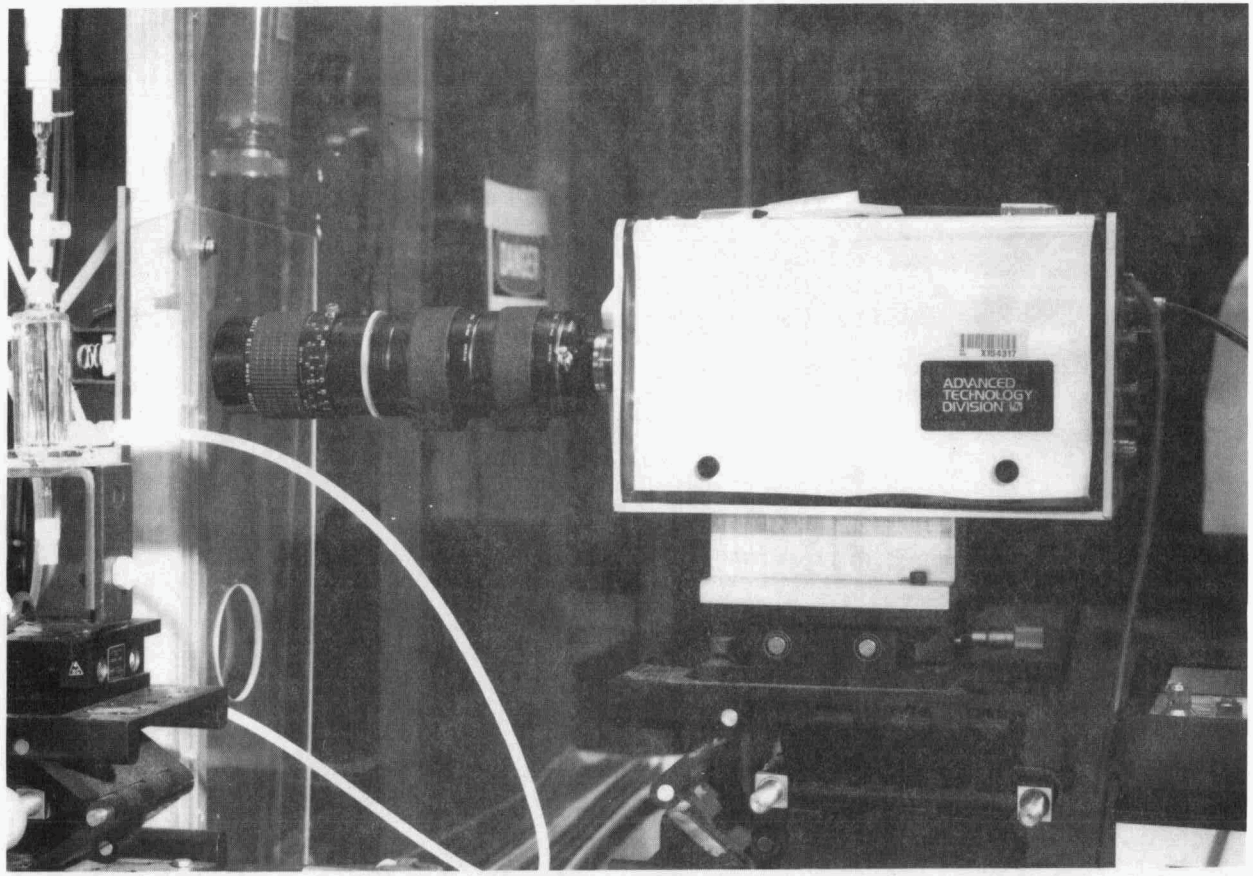

Fig. 12. Photograph of video camera, pendant drop column and syringe dispenser.

ORNL-Photo 2436-87

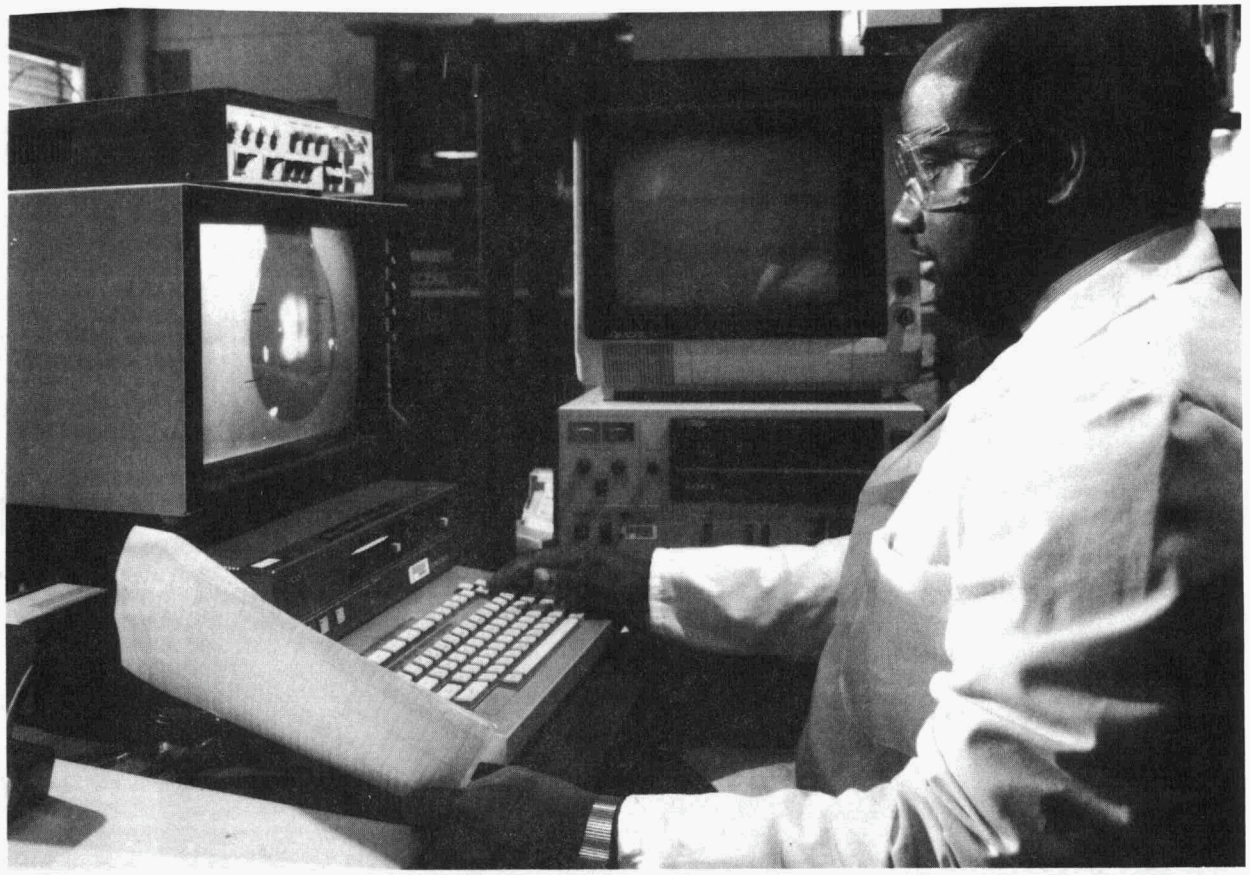

Fig. 13. Digitization of drop edge.

IBM AT computer by using ASCOM, a communication package for the Sony, and Symphony's communication package. The data were converted by the IBM basic program RK4TR8 to a form that could be used by the IBM basic nonlinear parameter estimation programs, NONLI4.BAS and NONLI8.BAS. To speed the computation, data were transferred to a main frame computer, the VAX 780, where the nonlinear program was translated into the FORTRAN 77 language. The FORTRAN 77 version of the program is entitled PENDR.F.

Finally, data sets generated from the nonlinear parameter estimation programs were used to graph observed and calculated data and residual plots. A graphic program, GRP4.BAS, developed by Giorgio Carta (Howard, Carta, and Byers, 1987) for the IBM AT computer was employed to generate the graphs. Typical input and output datasets are given in Appendix B.

4.3 CHEMICALS AND MIXING PROCEDURES

The water was triple distilled and stored in a glass container that was scrupulously washed with detergent and thoroughly rinsed with distilled water, ethanol, and hexane. All glassware was thoroughly cleansed to avoid contamination. The 2–ethyl hexanol was an Analytical Reagent grade MCB Preiser sample with a purity of 99%. Ethylene glycol, glycerin, and 1–octanol were an Analytical Reagent grade Fisher Scientific sample with a purity of 99.9, 99.5 and, 99.9%, respectively. Cyclohexane was an Analytical Reagent. grade EM Scientific product with a purity of 99.9%.

Saturation of each phase for interfacial tension studies was accomplished by mixing the liquids at the desired temperature for one day. The phases were then separated at the same temperature. It is desirable to keep the tempera-

ture constant during the saturation, separation, and measurement stages so that the system is in thermal and mass equilibria during interfacial tension measurements.

Density data for saturated solutions were determined by using pycnometers and a Mettler balance. A list of these data is given in Appendix A. The temperature was held to within $\pm 0.1°C$.

5. DISCUSSION OF RESULTS

5.1 Overview of Research

The interfacial tension of several fluid–fluid systems were determined by the pendant drop method and analyzed using the computer algorithms that are given in Appendix C. Initially, tests were conducted with synthetic data from Fordham's tables to determine the numerical accuracy of the algorithm. Secondly, surface and interfacial tension values for water–air and water–cyclohexane were obtained to test the overall accuracy of the experimental procedure and computations by comparing values from this study with literature values.

Several liquid–liquid systems that are important to electric field studies were also studied to provide data on the effect of temperature and surfactant concentration, i.e., NaCl, on interfacial tension. The main thrust of these studies was to provide interfacial tension data for systems with different electrical and chemical properties and to provide a technique for measuring the dynamic behavior of interfacial properties, e.g., interfacial tension.

5.2 Choosing Initial Parameter Values

Choosing the initial parameter estimates is essential to the convergence of the algorithm. The program RK4TR8 converts x, y coordinates from the transferred Sony file into a file that compensates for difference in x_s–y_s scaling and that has been corrected by the coordinates at the apex (x_o, y_o). Values for x_o, y_o and r_{xy} (the ratio of x_s per cm to y_s per cm) must be inserted by the user. Values for x_o and y_o are printed out when the graphic picture is being digitized by the Sony basic program, PFIND4. Since PFIND4 scans across the screen at a given y_s location, x_o is computed as follows:

$$x_o = \frac{\sum_{s=1}^{N}(x_{sl} + x_{sr})}{2N}, \tag{100}$$

where x_{sl} and x_{sr} are the x_s coordinates of left and right edges of the drop, respectively. The value y_o is the value of y_s at the bottom of the drop. If a large number of profile data points are available, the program RK4TR8 will fit the bottom data points to a quadratic expression and will compute the minimum of this function, which should give x_o and $y_o \times r_{xy}$ values that are close to the values given by PFIND4. Initial values of R_o and β_c are determined by selecting R_o to be $0.95 \times x_{max}$, i.e., $0.95 \times \frac{D_e}{2}$. The initial value for the shape factor, β_c, is $\frac{0.3}{R_o^2}$. Methods similar to Andreas and Hauser (1938) may have also been employed to obtain initial parameter estimates; however, such precise initial estimates were not required.

5.3 Computational Time and Accuracy

An NLPE program that was written in Microsoft BASIC for the IBM personal computer was initially used to determine surface and interfacial tension values. The software package, BASCOM compiler, was subsequently used to translate this program into a compiled code. A five-fold increase in speed was obtained. Using 50 data points, computational times were approximately 2.5 and 8 min per iteration for two–parameter (i.e., R_o and β_c) and four–parameter models (i.e., X_o, Y_o, R_o, and β_c), respectively, when numerical derivatives were used to compute the **Z**-matrix. A shorter time per iteration was achieved if analytical derivatives were used. With reasonable initial guesses for the parameters, the algorithm converged in four to ten iterations when numerical derivatives were used to compute the **Z**-matrix. When analytical derivatives were employed, the algorithm took longer to converge and had a tendency not to converge to the global minimum; but it would oscillate between two apparent local minima. The long computational time on the IBM AT computer made this method very unattractive. Therefore, the program which uses numerical derivatives was translated into the FORTRAN 77 language and ported to a compiler on the ORNL VAX 780 computer. A 30–fold increase in speed was obtained. The program (PENDR.F) now requires 6 min to converge if 120 data points are used. Since thirty to fifty data points should be adequate to achieve a precision of less than one percent, it will require approximately 1.5 min for convergence All computer programs are listed in Appendix C along with descriptions of the programs.

Another method of decreasing the computational time is to increase the

step size for the numerical integration routine. However, this will decrease the accuracy of the results. Following the experience of previous investigators (Girault, Schifflin and Smith 1982), a step size of $0.01 \times R_o$ was chosen. This step size is also approximately equal to the resolution of the digitization process.

Several tests were performed using synthetic profile data sets to evaluate the accuracy of the algorithm for determining interfacial tension. The data were taken from Fordham's tables where R_o and β have values of 1.000000, and -0.45, respectively. These data, i.e., x and y, have a resolution of $10^{-6} \times R_o$, and are stored in the data file BASAD2T. Another data file, BASAD, was constructed by truncating the values after the third decimal place; thus, resulting in a resolution of $0.01 \times R_o$. This is the approximate resolution achieved by digitization with the Sony Graphics Computer where R_o has an order of magnitude of 100 units.

Values of R_o and β_c are calculated from both data sets by the programs NONLI4.BAS, NONLI8.BAS, and PENDR.F. The programs NONLI4.BAS and NONLI8.BAS utilize, respectively, numerical, and analytical derivatives to form the **Z**− matrix; thus, a comparison between using numerical and analytical derivatives for determining parameter estimates was obtained. Table 1 gives the actual and calculated values of R_o and β, for the two synthetic data sets. It is observed that, as the resolution of the coordinates is decreased from $10^{-6} \times R_o$ to $0.01 \times R_o$, the percent difference between the calculated and expected shape factor, β, increases from 10^{-3} to 0.5%. This would also mean a similar error in calculating the surface or interfacial tension. It is, therefore, concluded that an accuracy and precision (for digitization) of approximately 0.5% can be expected with the present digitization technique and with the use of either numerical or analytical derivatives to determine the **Z**-matrix.

Table 1. Test of algorithms on Fordham's data set

$(R_o = 1.000000, \beta = -0.45)$

Program name	Resolution (r_s)	Calculated values (R_o)	(β)	Difference (%) (R_o)	(β)
NONLI4.BAS	1E-6	0.99993	-0.4502	0.04	0.07
NONLI4.BAS	0.01	1.00100	-0.4480	0.10	0.44
NONLI8.BAS	1E-6	0.99998	-0.45001	0.0024	0.0020
NONLI8.BAS	0.01	1.00100	-0.4480	0.10	0.44
PENDR.F	1E-6	0.99998	-0.450003	0.0016	0.0007
PENDR.F	0.01	1.00100	-0.4470	0.10	0.45

Figs. 14 and 15 illustrate the results from the curvefitting of the synthetic data sets (BASAD and BASAD2T, respectively) by the algorithm. Furthermore, residual plots for each data set are given in Figs. 16 and 17. The residual plots are constructed by subtracting x_{obs} from x_{calc} and plotting these values against y_{obs}. Residuals for the BASAD data set exhibits a random distribution about zero and do not exceed about 0.005 units. However, the residuals for BASAD2T, although very small ($< 10^{-4}$), show a distinct pattern as a function of y. This is most probably due to the interval of integration being much greater than the resolution of the data. A further investigation of the residuals was not conducted, since practical data would have a resolution much greater than the $0.01 \times R_o$.

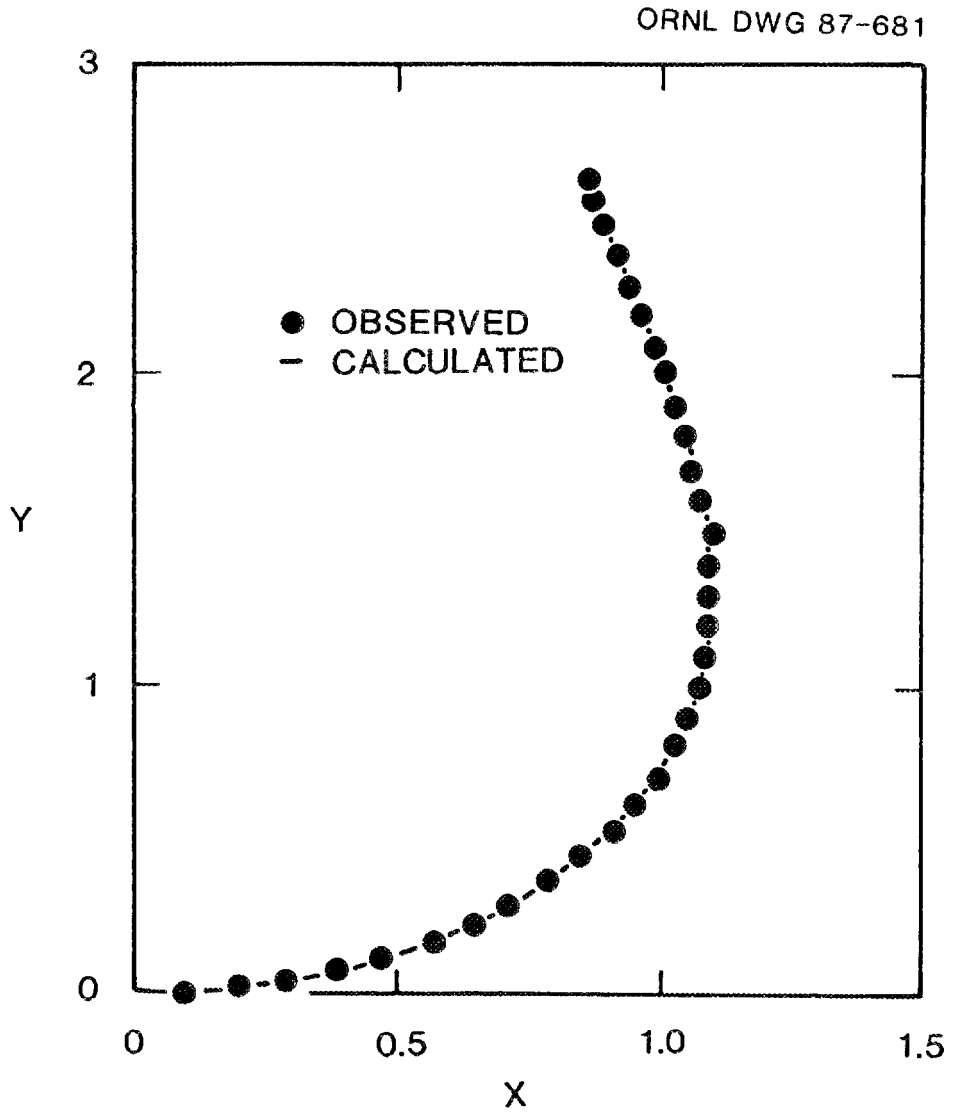

Fig. 14. Pendant drop profile: Fordham's data (BASAD), where X and Y are dimensionless horizontal and vertical coordinates.

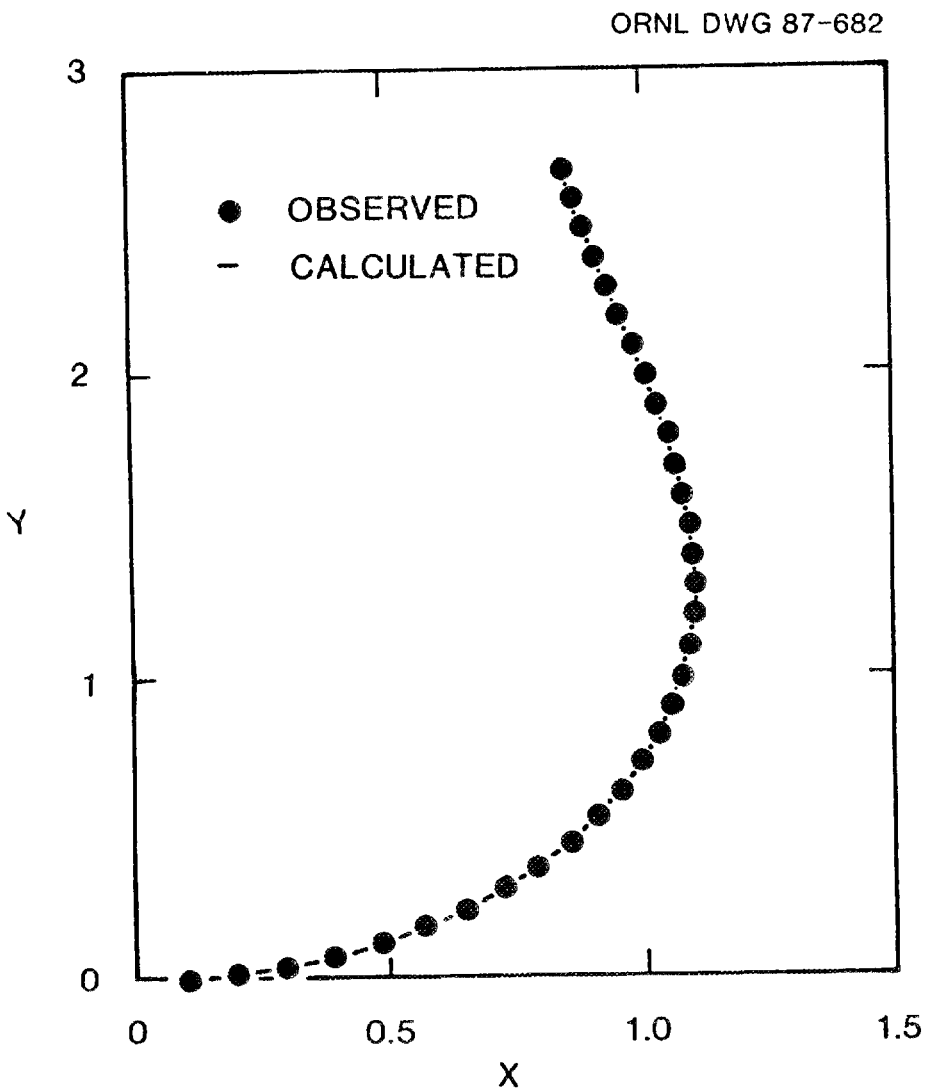

Fig. 15. Pendant drop profile: Fordham's data (BASAD2t), where X and Y are dimensionless horizontal and vertical coordinates.

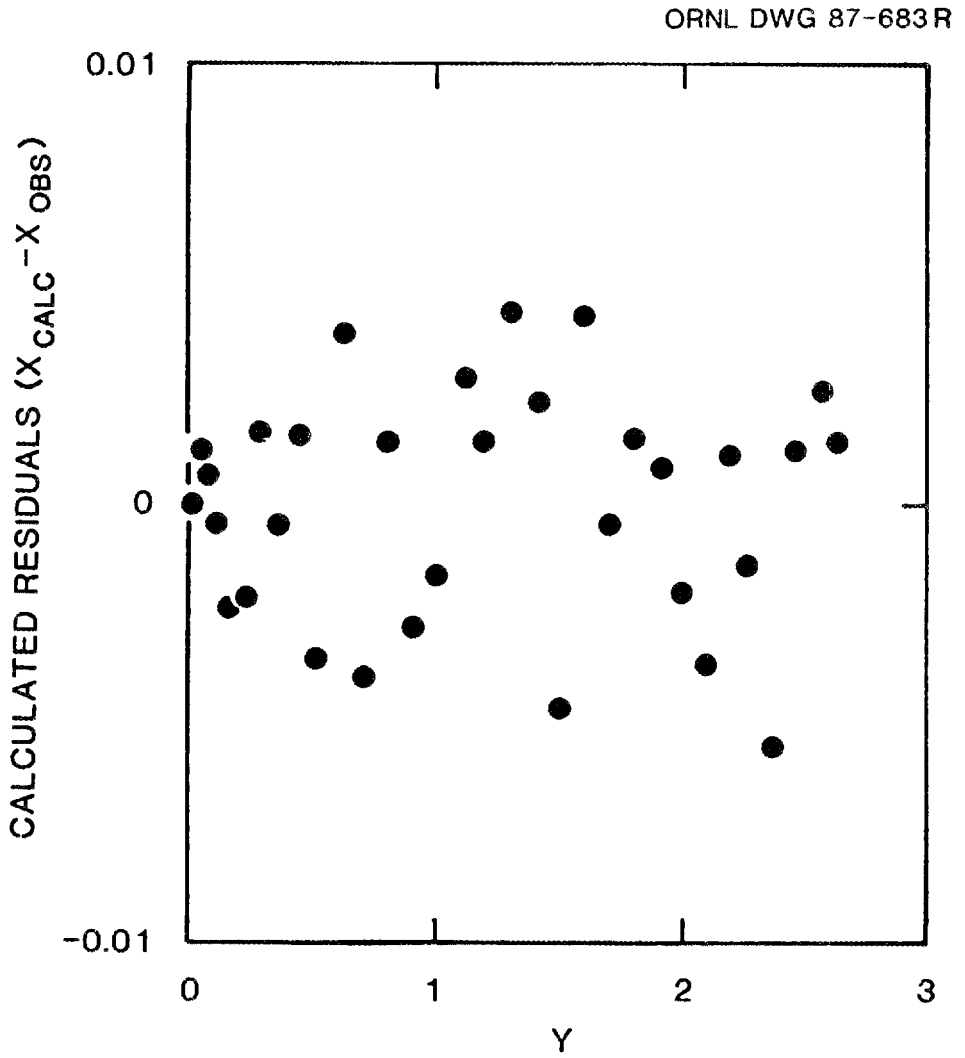

Fig. 16. Pendant drop profile: Residuals (BASAD), where X and Y are dimensionless horizontal and vertical coordinates.

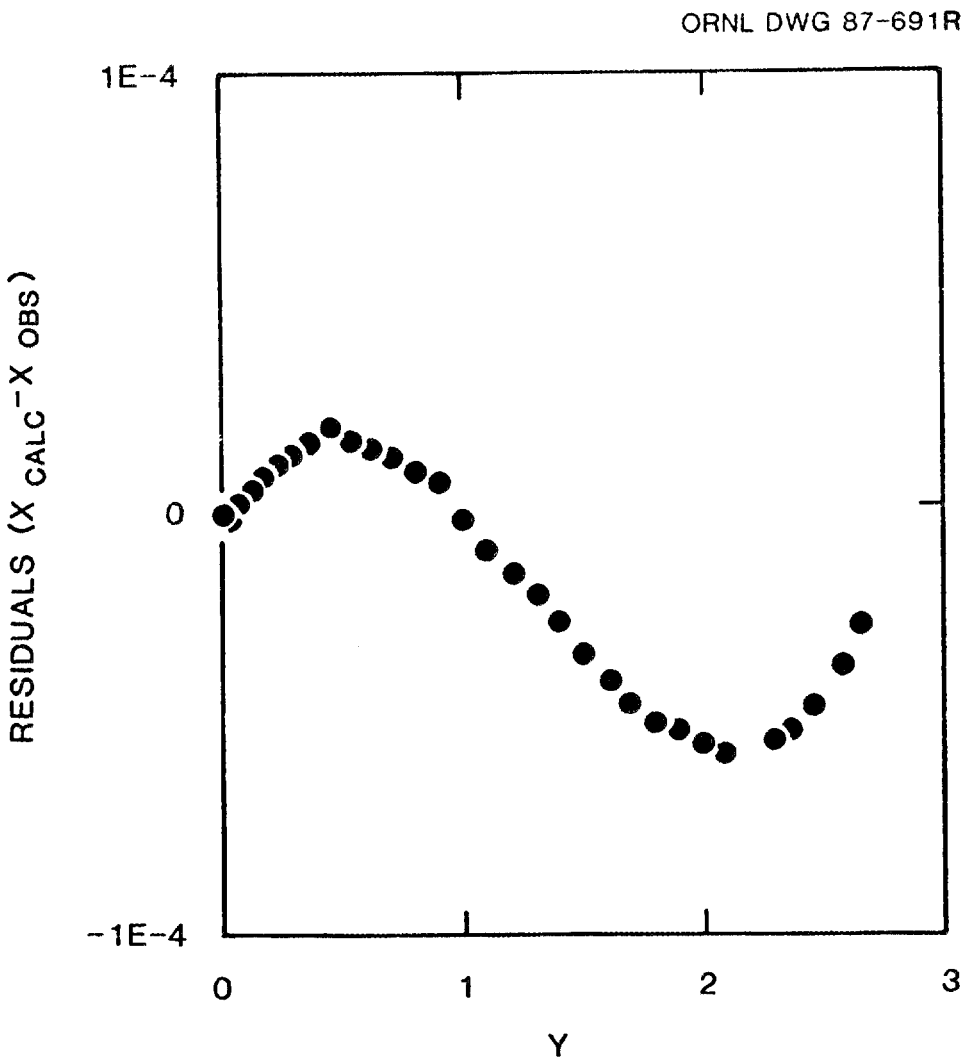

Fig. 17. Pendant drop profile: Residuals (BASAD2t), where X and Y are dimensionless horizontal and vertical coordinates.

5.4 Interfacial Tension of Test Systems.

Initial experiments were done to determine the surface and interfacial tension of systems that would show the comparison between experimental values with literature values. The surface tension of water in air was initially determined at several temperatures by the two–parameter algorithm. Three drops were traced and surface tension was determined from each trace. Surface tension values had an average standard deviation of 1.1 dyne/cm (11μ N/cm) or a standard error of the mean of approximately 1.5%. The values are given in Table 2 along with literature data. The percent error between our values and literature values (Hodgman 1951) averaged 0.14%. Using the four–parameter model on a separate data set, a value of 72.2 dyne/cm (720 μN/cm) was obtained for the surface tension of water in air at $21.0°C$. A value of 72.6 was interpolated from data given by Hodgman 1951 and, therefore, a difference of 0.55% was obtained. Plots of the experimental and theoretical curve, and residuals are shown in Figs. 18 and 19. The fit of the experimental data is very good; however, there is a pattern in the residuals plot which tends to indicate that the drop was slightly tilted. This will cause a small error in estimating the surface tension. However, the residuals are generally smaller than the resolution for digitization; and it is, therefore, expected that this error should be smaller than 1%.

An interfacial tension of 50.3 ± 1.2 ($\pm$ 2.5%) dyne/cm (503 μN/cm) (Table 2) was obtained for water–cyclohexane at $20°C$ using the two–parameter model and 50.9 ± 1.1 (± 2.2) dyne/cm (509μN/cm) using the four–parameter model. Values of 50.2 dyne/cm and 51 dyne/cm (Petre 1966) were reported for the interfacial tension of water–cyclohexane at $20°C$. Boyce et al. (1984) reported

Table 2. Surface and interfacial tension of fluid-fluid systems

System	Temperature (^oC)	Surface and interfacial tension (dyne/cm) Experimental	Literature[a]
Water–air	25.0	72.0 ± 1.0	71.97
Water–air	28.8	71.3 ± 1.2	71.3
Water–air	30.8	70.7 ± 1.2	70.95
Water–Cyclohexane	23.5	50.2 ± 1.7	50.2[b]
Water–1-octanol	23.5	8.5 ± 0.2	8.5[b]

[a] Literature values interpolated from data presented in The Handbook of Chemistry and Physics, 33rd edition, 1951-1952.

[b] Value at 20^oC reported by Good, 1970.

that the algorithm by Rotenberg et al. (1981) resulted in standard errors of the mean ranging from 1.2 to 3.4% for systems with interfacial tensions ranging from 5.7 to 38.4 dyne/cm. Therefore, the standard errors of the mean observed in the preliminary test of our technique are similar to those obtained by Rotenberg's algorithm.

With these preliminary results, it was concluded that the algorithm accurately estimated surface and interfacial tension; however, changes in the experimental setup were required to improve the precision.

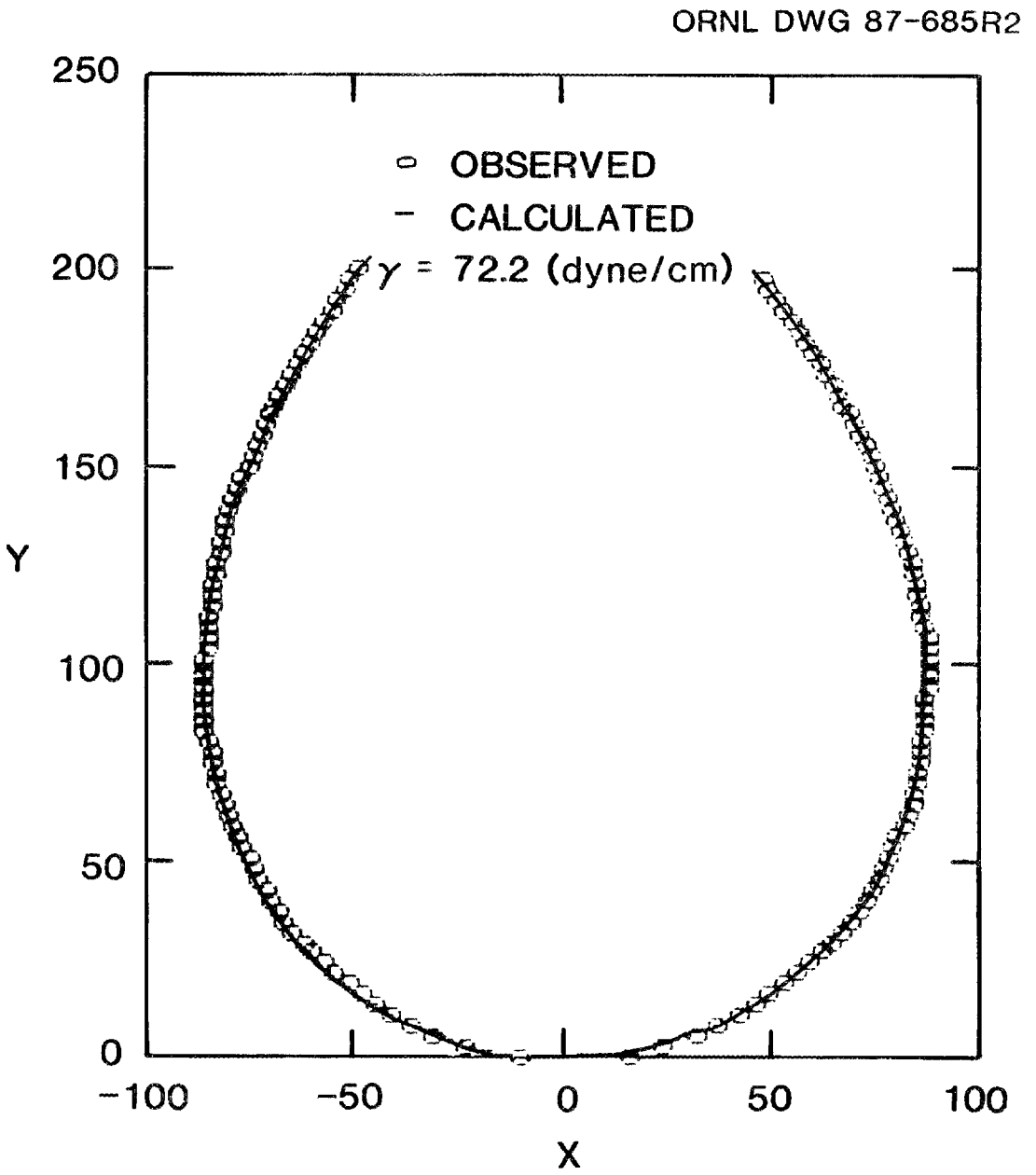

Fig. 18. Curvefitting experimental pendant drop profile data to the Laplace equation: Water-Air System (21° C)

ORNL DWG 87-684R

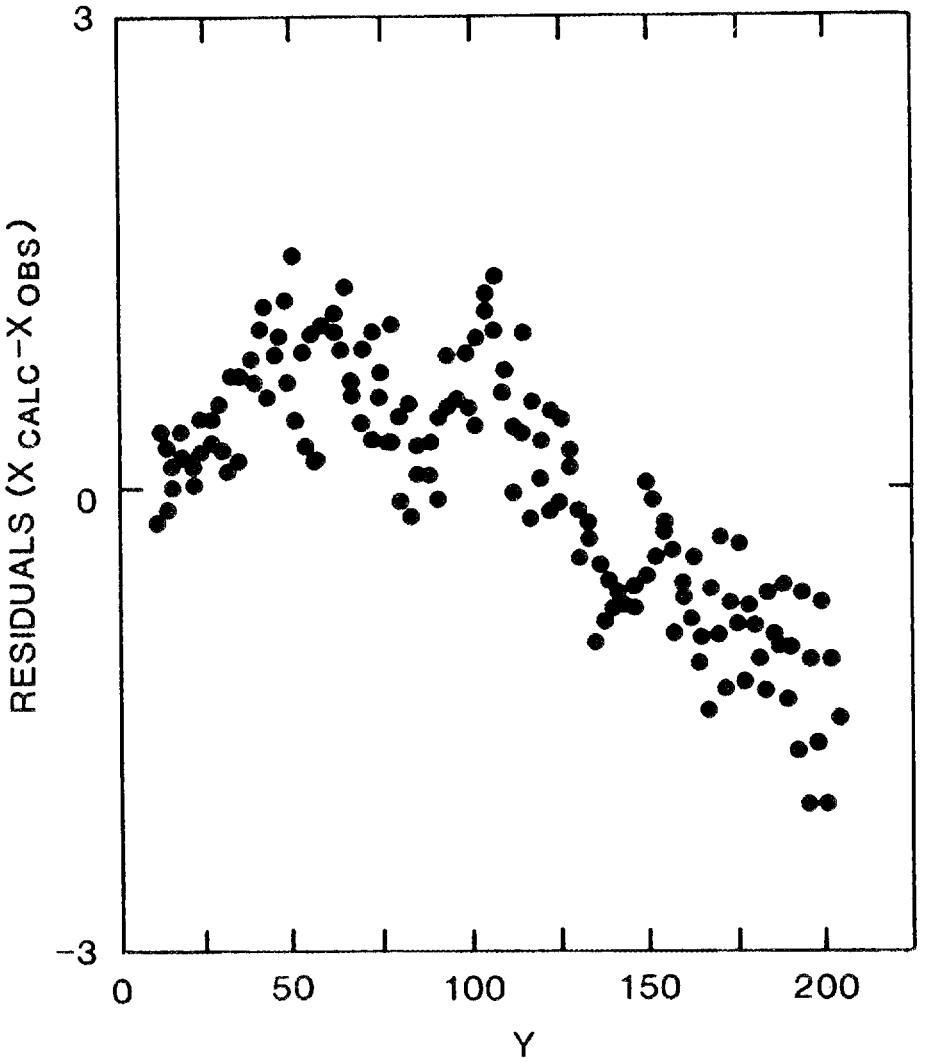

Fig. 19. Curvefitting experimental pendant drop profile data to the Laplace equation: Residuals (Water-Air System)

5.5 Interfacial Tension of Other Systems

Table 2 also gives the interfacial tension of water–octanol. The water–octanol data were obtained in the same experimental device as the water–air and water–cyclohexane systems. Interfacial tensions of 8.6 ± 0.1 ($\pm 1.2\%$) and 8.5 ± 0.1 ($\pm 1.2\%$) dyne/cm (86 and 85 μN/cm) at $23.4°C$ were obtained for this system using the two and four parameter models, respectively. This agrees with a literature value of 8.5 dyne/cm ($25°C$) that was given in Good and Elbing (1970). It should be mentioned that great care was taken to accurately determine x_o and y_o from the digitized data of the first three systems (water–air,water–cyclohexane, and water–octanol). Therefore, surface and interfacial tension values as calculated by the two and four parameter models were in very close agreement (i.e., $< 1\%$ difference).

A modified experimental setup was used to determine the interfacial tension of the systems given below. This setup allowed for better temperature control. Furthermore, the image of the pendant drop was magnified by inserting a longer barrel for the camera lens. This allowed focusing on a smaller portion of the glass column, which would decrease any nonlinear scaling of the x and y axis due to distortion in curvature of the glass tubing and would also slightly improve the resolution of the digitizer. As a result of this change, the image of the drop was enlarged on the video screen. Consequently, the array for storing the drop edge data point at every y_s position was too large for the memory of the Sony computer. The graphics picture was, therefore, digitized by scanning the edge at every other y_s position. This meant that the y_o that was determined by PFIND4 may be in error by at most one division. As is shown below, this uncertainty in

y_o may cause substantial error in determining interfacial tension.

The interfacial tensions of water–2-ethyl hexanol, glycol–cyclohexane, and glycerin–cyclohexane as a function of temperature were determined by the two–parameter and four–parameter models. These data are given in Table 3 and were obtained from the modified experimental setup that was given in Section 4.2. The difference between interfacial tension values calculated by the two methods shows the consequence of using the two parameter model when there is uncertainty in y_o. Percent differences from 1% to 10% were observed for interfacial tension values that were determined by two and four parameter models.

Evaluating the water–2-ethyl hexanol data, we see that the values from the two–parameter model show a minimum in the interfacial tension as the temperature was increased from 22.4 to 35.3° C. The behavior of the interfacial tension for water–long chain alcohol suggest, however, that the interfacial tension should increase with increase in temperature (Aveyard and Saleem 1977); and furthermore, interfacial tension should increase linearly with temperature. Using the four–parameter model, this behavior (Fig. 20) is observed and the entropy of interface formation, $-\frac{d\gamma}{dT}$, is −0.08 dyne/cm/° C (−0.8 μN/cm/° C). The interfacial tension of 13.7 dyne/cm (137 μN/cm) for water–2-ethyl hexanol at 22.4° C that is reported in this study confirms a value of 13.9 dyne/cm (139 $\mu N/cm$) (25° C) reported by Clinton (1968).

Fig. 20 also shows the temperature dependence for the interfacial tension of glycol–cyclohexane. The entropy of interface formation is 0.064 dyne/cm 0.64 μN/cm). Literature data supports this trend. The interfacial tension of glycol–cyclohexane is reported as 14.4 dyne/cm (144 μN/cm) and 14.2 dyne/cm (142 μN/cm) at 20 and 40° C, respectively (Timmermans 1959).

Table 3. Interfacial tension of liquid-liquid systems

System	Temp. (° C)	Interfacial tension, γ (dyne/cm)[a] Two parameter model	Four parameter model
Water-2-ethyl:			
hexanol	22.4	13.2 ± 0.2	13.7 ± 0.2
hexanol	30.4	13.0 ± 0.1	14.2 ± 0.1
hexanol	35.3	13.6 ± 0.6	14.8 ± 0.2
Glycol:			
cyclohexane	21.1	15.2 ± 0.0	15.4 ± 0.0
cyclohexane	29.4	14.5 ± 0.1	14.8 ± 0.1
cyclohexane	(33.8)	14.0 ± 0.1	14.6 ± 0.1
Glycerin:			
cyclohexane	20.3	23.8 ± 0.5	24.6 ± 0.5
cyclohexane	28.8	24.0 ± 0.1	24.4 ± 0.1
cyclohexane	34.3	24.3 ± 0.3	24.8 ± 0.3

[a] 1 dyne/cm = 10 μN/cm.

The 6.9% difference between the interfacial tension that is reported in this study [(15.4 dyne/cm)154μN/cm] and the literature value of 14.4 dyne/cm (144 μN/cm) cannot be fully evaluated because the previous workers did not

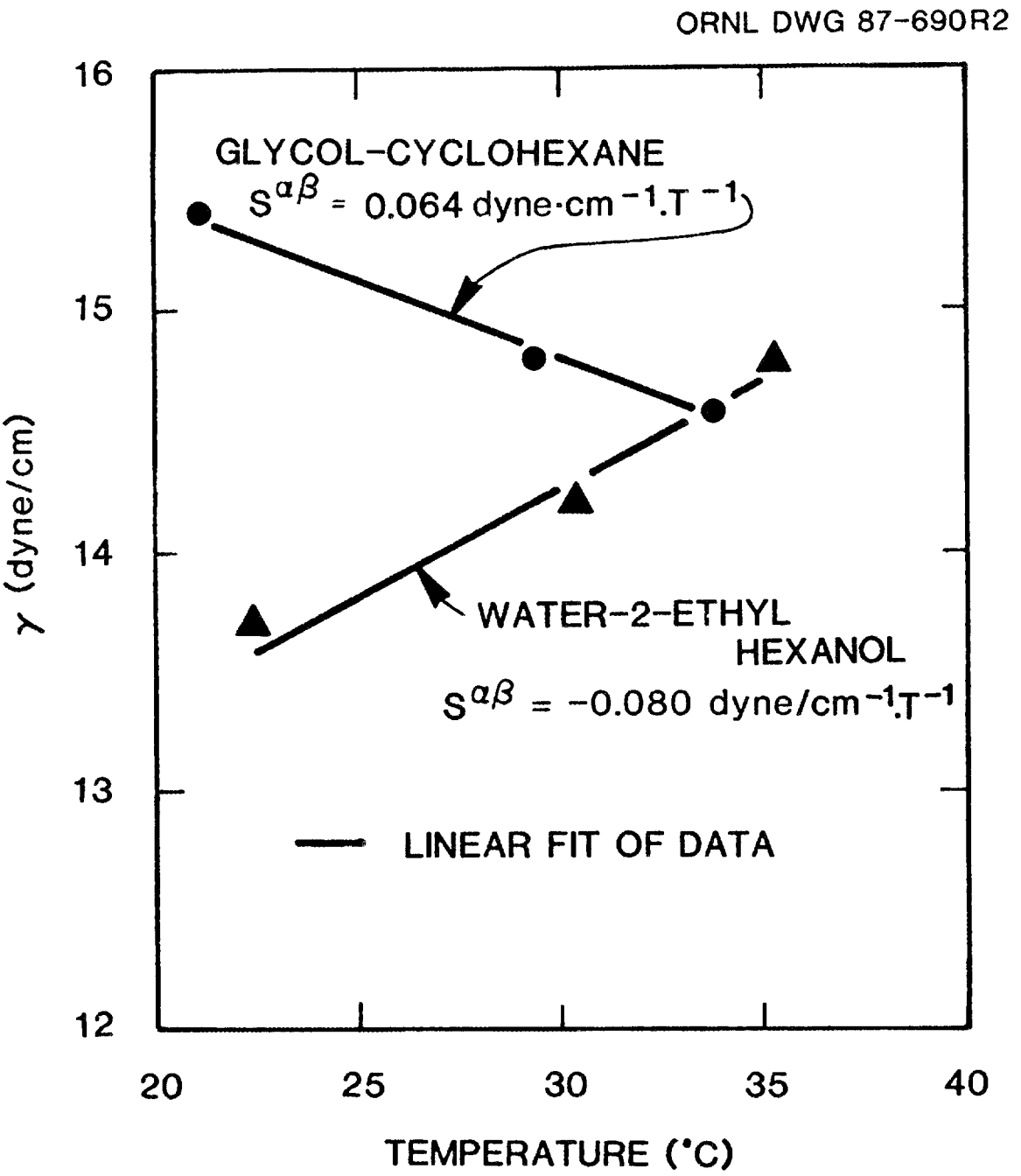

Fig. 20. Effect of temperature on interfacial tension

give the method for determining this value or the purity of the solvents.

The interfacial tension of glycerin–cyclohexane did not appear to vary with temperature and averaged 24.6 dyne/cm (246 μN/cm) over the temperature range 20.3 to 34.3°C. No literature data was found on this system. However, as will be seen in a subsequent section, this value does agree with the interfacial tension value that is predicted by Fowkes equation.

5.6 Effect of Salt on Interfacial Tension

Surfactant will affect the interfacial tension of a liquid–liquid system in several ways depending on the manner in which the surfactant absorbs in the bulk phases and interfacial region. Eq. 64 may be used for strong adsorption at the interface of a solute from dilute solutions. It is seen that the term Γ_3^s is positive and, therefore, the interfacial tension will decrease with an increase in bulk surfactant concentration. If the surfactant absorbs more into a bulk phase the interfacial tension may increase. This can be seen if we begin with Eq. 64. Equation 64 reduces to Eq. 101 if the following reasonable assumptions are used:

1. The solvents 1 and 2 are immiscible, i.e. $x_1^\beta = 0$, $x_2^\alpha = 0$.
2. The surfactant, solute, is soluble in only one of the phases, i.e. $x_3^\beta = 0$.
3. The solute does not adsorb strongly at the interface, i.e. $\Gamma_3^s = 0$.

Then

$$d\gamma = \frac{x_3^\alpha}{x_1^\alpha}\Gamma_1^s\, d\mu_3. \tag{101}$$

This equation suggests that the interfacial tension should increase with an increase in the chemical potential or solute concentration.

Fig. 21 shows the effect of salt concentration on the interfacial tension of water–2-ethyl hexanol. This system is of practical importance in that the electrical properties of the water–2-ethyl hexanol is changed by adding salt to the system. Table 4 gives the interfacial tension and electrical conductivity data for this system. The interfacial tension increases with an increase in salt concentration. Furthermore, the interfacial tension increases linearly with salt concentration in the region $x_3^\alpha = 0.1001$ to 0.3165 g/mL. This linear behavior

can be explained if the chemical potential of the salt in the aqueous phase can be expressed by the equation:

$$d\mu_3 = RT[d\ln(a_c x_3^\alpha)], \tag{102}$$

where a_c is the activity coefficient. If it is assumed that for dilute solutions a_c and x_1^α are 1.0, then substitution of Eq. 101 into Eq. 102, the following equation is obtained:

$$d\gamma = \frac{\Gamma_1^s RT}{1} dx_3. \tag{103}$$

Upon integrating this equation, we can see that the interfacial tension increases linearly with an increase in salt concentration.

Table 4. Effect of Salt Concentration on Interfacial Tension of Water–2-ethyl hexanol.

Concentration of salt in H_2O (g/mL)	Conductivity[a] $(\text{ohms})^{-1}$	Interfacial tension[b] (dyne/cm)
0.3165	400	21.7 ± 0.07
0.1700	270	16.6 ± 0.20
0.1001	118	14.2 ± 0.08
0.0000	0	13.7 ± 0.2

[a] 1 ohm^{-1} = 1 siemens (S)

[b] 1 dyne/cm = 10 μN/cm

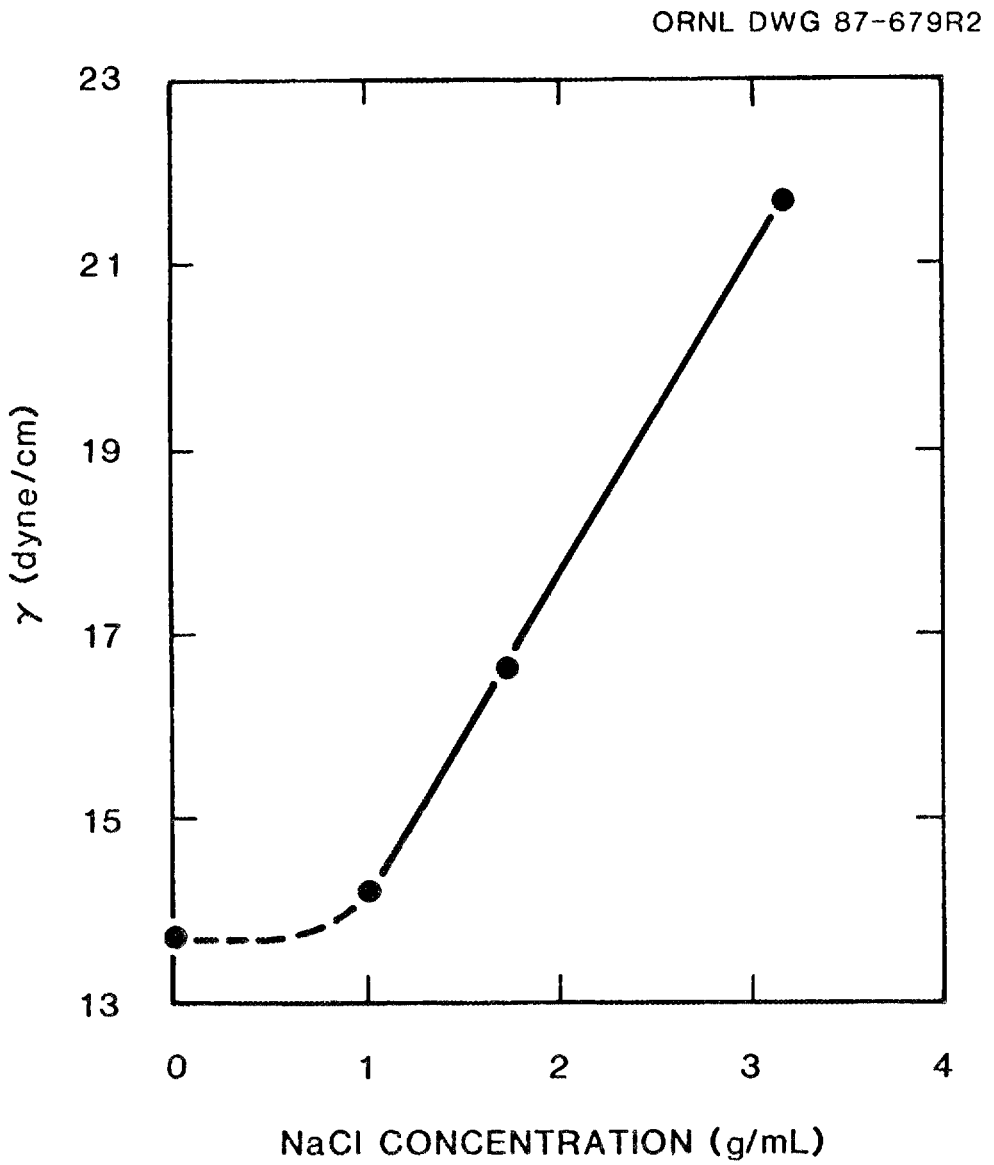

Fig. 21. Effect of aqueous phase salt concentration on the interfacial tension of water–2-ethyl hexanol.

Fig. 21 also illustrates that as the concentration of salt in the aqueous approaches zero, the slope of the curve changes. This is explained by the fact that as x_3^α approaches zero, the above assumptions 1 and 2 are no longer valid; and also the assumptions that x_1^β and $x_2^\alpha = 0$ is also invalid, since x_1^β and x_2^α approaches the same order of magnitude as x_3^α as x_3^α —-> 0. Equation 63 must, therefore, be employed to evaluate solute effects on interfacial tension at very low salt concentrations.

5.7 Estimating Interfacial Tension

In many cases, the interfacial tension for a particular system is not available; and it becomes necessary to estimate this physical property from existing data such as surface tension, solubility, molecular weight, and molecular structure. Table 5 gives interfacial tension estimates for (1) water–2-ethyl hexanol, (2) glycol–cyclohexane, and (3) glycerin–cyclohexane. Methods by Good and Elbing (1970), Fowkes (1964), Donahue and Bartell (1952), and Gaines (1984) were employed to obtain interfacial tension estimates. All of these methods, with the exception of Gaines, have been discussed in the theory section. Gaines presents the following empirical linear equation for the homologous series, diol and alkanes:

$$\gamma_{12} = -5.03 + 408.2 M_{diol}^{-2/3} - 82.38 M_{alkane}^{-2/3}, \tag{104}$$

where M is molecular weight.

Although this equation is for alkanes/diols, it has been applied to water–alkane systems by replacing the molecular weight of the diol with the molecular

Table 5. Estimating interfacial tension by various methods

	Interfacial tension (dyne/cm)					
System	This work	Fowkes	Good		Donahue	Gaines
Water–2-ethyl hexanol:	13.7	N/A	11.3[a]	10.1[b]	7.6	N/A
Glycol–Cyclo-hexane:	15.4	N/A	N/A	N/A	N/A	16.7
Glycerin–Cyclo-hexane:	24.6	27.5	N/A	N/A	N/A	N/A

N/A – Not applicable or not available [a] Assumes solubility effects are negligible and $\Phi = 1.0$.

[b] Includes solubility effects and uses an interpolated value of 0.96 for Φ.

weight of water. Excellent agreement between experimental and calculated interfacial tensions for water/alkane and diol/alkane systems was obtained. It is also expected that good estimates for cyclohexane–ethyl glycol should also be obtained from this equation. The calculated interfacial tension of glycol–cyclohexane is 16.7 dyne/cm (167 μN/cm), which agrees to within 10% of the experimental value of 15.4 dyne/cm (154 μN/cm) that was obtained in this study.

The estimate for the interfacial tension of water-2–ethyl hexanol was cal-

culated by the expressions given by Good and Ebling (1970) and Donahue and Bartell (1952). Value (a), 11.3 dyne/cm (113 μN/cm), was determined from Eq. 20, where Φ was assumed to be 1.0 and solubility effects were neglected. A value of 10.1 was obtained if solubility effects were taken into account and a value, Φ_s, of 0.96 is used. The value for Φ_s was interpolated from data values given by Good for substances with a range of solubilities. Donahue's equation yielded an interfacial tension of 7.6 dyne/cm (76 μN/cm), which is considerably lower than the experimental value of 13.7 dyne/cm (137 μN/cm). The large differences between the calculated and experimental data shows the need for developing better correlations for the estimation of interfacial tension.

Interfacial tension for the glycerin–cyclohexane system was determined by the Fowkes equation, which may be applied to binary systems where the intermolecular forces in at least one of the liquids are predominately due to dispersion forces. Furthermore, the adhesive forces between the two liquids must only be due to dispersion forces. To use this equation, it is assumed that the dispersion force component of the surface tension is known for the second liquid. Fowkes (1964) reported a dispersion force–surface tension component of 37.0 ± 4.0 dyne/cm (370 μN/cm), which was determined by contact angle measurements between glycerol and a solid hydrocarbon. Inserting this value into Fowkes equation and using the literature value (Scheflans and Jacobs 1953) of $\gamma_{glycerin} = 63$ dyne/cm (630 μN/cm), an interfacial tension of 27.5 ± 3 dyne/cm (275 μN/cm) was obtained. This agrees reasonably well with the experimental value of 24.6 dyne/cm (246 μN/cm).

Using Fowkes equation, the dispersion force–component for the interfacial tension, γ^d, of glycerol and glycol can be estimated from the experimental in-

terfacial tensions for glycerol–cyclohexane and glycol–cyclohexane given in the present study and pure component interfacial tensions. Values of 40.6 and 29.5 dyne/cm (406 and 295 μN/cm) were computed for glycerol–cyclohexane and glycol–cyclohexane systems, respectively. With use of Fowkes equation, these values will be helpful in predicting the interfacial tension between these polyhydric compounds and other alkanes.

In summary, a generalized correlation that accurately computes interfacial tension has not been developed. The method by Good has the best potential of becoming an excellent generalized correlation. This method has a strong theoretical basis and is attractive in that pure component surface tensions and solubility data may be used. Pure component surface tensions can be calculated from the methods by MacLeod and Sugden, Grain, Walden, and Mayer (Grain 1982). The major drawback in using this method is estimating Φ. Reliable values have only been calculated for hydrocarbon-water systems; when applied to hydrogen-bonded systems such as alcohol-water, calculated values of Φ do not give accurate results. Even when Φ is interpolated from existing experimental values for similar systems large errors may be obtained. However, the development of empirical correlations for the parameter Φ (for the various classes and structure of compounds) and solubility could make this method more attractive. The other methods for estimating interfacial tension are, at best, empirical and applicable only to specific liquid-liquid systems.

6. CONCLUSIONS AND RECOMMENDATIONS

Interfacial tension can be accurately determined from drop edge coordinates of pendant drops by the nonlinear parameter estimation algorithm that was

developed in this study. The overall accuracy and precision of the method is determined by the degree of asymmetry of the drop coordinates, the ability to accurately determine the drop edge and apex coordinates (x_o and y_o), and the resolution of the digitization process. If x_o and y_o are known, the two parameters are employed to accurately determine the interfacial tension by varying the parameter β and R_o. However, for general use where the apex coordinates are not known, a four parameter model is used which varies the parameters x_o, y_o, β and R_o to minimize the objective function. Using test data from Fordham's tables, an error in interfacial tension of 0.5% was determined for data points which could be determined with an accuracy of $0.01 \times R_o$.

Initial parameter estimates of x_o, y_o, β, and R_o are easily obtained by properly digitizing the image such that reasonable estimates of x_o and y_o are determined. Subsequent computations are made to determine D_e, and estimates of R_o and β are obtained. With these initial estimates, the computer algorithm converges in approximately 10 iterations. Computational times on a VAX 780 are 6 min for 120 drop edge coordinates. For most applications, 30 to 50 data points are all that is required to achieved a precision of 1%, and therefore, convergence occurs in approximately 1.5 minutes.

Repetitive measurements of the interfacial tension for several fluid-fluid systems shows an overall precision, i.e, standard error about the mean, of 1.2 to 2.5% for digitization and computation. This agrees with values reported by Boyce et al. (1984) who used uses a similar technique for digitization and computation. The overall precision can be improved by increasing the resolution for digitization and ensuring that the drop is axisymmetrical.

In addition to developing the algorithm and digitization procedure, several

tests were conducted to evaluate the effect of several properties on the interfacial tension of systems that are important to electric field studies. The temperature dependence of the interfacial tension for several liquid–liquid systems shows a negative entropy for interface formation for systems where both phases contain liquids with polar function groups and a positive entropy for interface formation if one of the liquids does not contain a polar functional group. This agrees with the findings of Aveyard et al. (1977).

Varying the salt concentration in the aqueous phase of the water–2-ethyl hexanol system shows an increase in interfacial tension with increase in salt concentration. The linear increase in interfacial tension with salt concentration in the range, 0.1001 to 0.3165 g salt/mL is predicted by thermodynamic principles. The apparent deviation from linearity that is observed at lower salt concentrations is qualitatively explained thermodynamically.

Future studies in this area should include the use of an IBM PC/AT compatible high resolution (512x512 pixels) frame-grabber (Data Translation Inc., DT 2851-60Hz) for real–time digital image processing. The frame–grabber should be interfaced with a high speed 16-bit image processor (Data Translation Inc. DT 2858). The use of a higher resolution frame–grabber will minimize the error due to digitization. Improved methods for assuring the symmetry of the drop image about the y-axis are also needed if very accurate and precise measurements are to be achieved. This could be done by carefully machining an adjustable lid with an unremovable nozzle that is perpendicular to the horizontal plane. The lid could then be adjusted to achieve perfect verticality.

It would also be interesting to test PENDR.F on an IBM PC/AT computer which is equipped with a high speed co-processor (Definicon Systems Inc.(DSI-

780+/4). This board allows processing of FORTRAN at speeds comparable to mainframe computers. Furthermore, the user would be able to perform all the necessary computations on one system which would improve the overall time for obtaining interfacial tension values from pendant drops.

Studies should also be conducted to determine the effect of low salt concentrations on the interfacial tension of water–2-ethyl hexanol. Previous investigators have observed a minimum in the interfacial tension at low salt concentrations (Beunen and Ruckenstein 1982).

7. REFERENCES

Adamson, A. W. 1967. Physical Chemistry of Surfaces, John Wiley, New York.

Ambwani, D. S., and Fort, T., Jr. 1979. Surf. and Colloid Sci., 11, Plenum, New York.

Andreas, J. M., Hauser, E. A., and Tucker, W. B. 1938. J. Phys. Chem., 42, 1001.

Aveyard, R., Briscoe, B. J., and Chapman, J. 1972. J. Chem. Soc. Faraday I, 68, 10.

Aveyard, R., and Saleem, S. M. 1977. J. Chem. Soc., Faraday I, 73(6), 896-904.

Aveyard, R., and Vincent, B. 1977. Prog. in Sur. Sci. 8, 69-102.

Bashforth, F., and Adams, J. C. 1883. An Attempt to Test the Theories of Capillary Action, Cambridge Univ. Press, Cambridge, Mass.

Boyce, J. F., Schurch, S., Rotenberg, Y., and Newmann, A. W. 1984. Colloids and Surf. 9, 307-317.

Bridger, K. 1977. "Vax Fourtran Program for Solving the Surface Tension of an Axisymmetric Drop," Martin Marietta Research Laboratory Research Report, Baltimore, Md.

Buenen, J. A., and Ruckenstein, E. 1982. Adv. in Colloid and Interface Sci., 16, 201-231.

Butler, E. D. 1967. J. of Phys. Chem., 67, 1419.

Clinton, S. D. 1968. Mass Transfer of Water from Single Thoria Sol Droplets Fluidized in 2-Ethyl-1-Hexanol, ORNL-TM-2163.

Donahue, J. D., and Bartell, F. E. 1952. J. Phys. Chem., 56, 480.

Draper, N. R., and Smith, H. 1981. Applied Regression Analysis, John Wiley and Sons.

Fordham, S. 1948. Proc. Roy. Soc. (London), 194 A, 1-16.

Fowkes, F. W. 1964. Ind. Eng. Chem., 56, 12, 40.

Gaines, G. L. 1978. J. of Colloid and Inter. Sci., 66, 3, 593-594.

Gaines, G. L., Jr. 1984. Colloids and Surf., 8, 383-389.

Girault, H. H., Schiffrin, D. F., and Smith, B. D. V. 1982. J. Electroanal. Chem., 137, 207-217.

Girault, H. H., and Schiffrin, D. F. 1983. J. Electroanal. Chem., 150, 43-49.

Girifalco, L. A., and Good, R. J. 1957. J. Phys. Chem., 61, 904.

Good, R. J., and Elbing, E. 1970. Ind. and Eng. Chem., 62, 54.

Grain, C. F. 1982. Handbook of Chemical Property Estimation Methods, eds. W. J. Lyman, W. F. Reehl, and D. H. Rosenblatt, McGraw-Hill Book Company, New York.

Guggenheim, E. A., 1967. Thermodynamics, North-Holland Publishing Co., Amsterdam.

Hiemenz, P. C. 1986. Principles of Colloid and Surface Chemistry, Marcel and Dekker, Inc., New York.

Hodgman, C. D. 1951. Handbook of Chemistry and Physics, 33rd Edition, Chemical Rubber Corporation, Cleveland, Ohio.

Howard, A. J., Carta, G., and Byers, C. H. 1987. Separation of Sugars by Continuous Annular Chromatography, ORNL/TM-10318.

Huh, C., and Reed, R. L. 1983. J. of Colloid and Inter. Sci., 91, 2, 472.

Jaycock, M. J., and Parfitt, G. D. 1981. Chemistry of Interfaces, John Wiley and Sons Inc.
Jennings, H. Y. 1957. Rev. of Sci. Inst., 28, 10, 774.
Lyons, C. J., Elbing, E., and Wilson, I. R. 1985. J. Chem. Soc. Faraday Trans. 2, 81, 327-339.
Malinauskas, R. A., and Byers, C. H. 1985. Drop Formation Phenomena in DC Electric Fields, ORNL/TM-9446.
Maze, C., and Burnet, G. 1969. Sur. Sci. 13, 451.
Maze, C., and Burnet, G. 1971. Surf. Sci., 24, 335.
Niederhauser, D. O., and Bartell, F. E. 1950. Report of Progress in Fundamental Research on Occurrence and Recovery of Petroleum 1948–1949, American Petroleum Institute, Baltimore, Md., 114-146.
Padday, J. F. 1969. Surf. and Colloid Sci., 1, 103.
Petre, G., and Schayer–Polischuk, M. L. 1966. J. de Chim. Phys. et de Physico –Chimie Biol., 63, 1409-1415.
Ramakrishnan, S., Princq, and Hartland, S. Ind. J. of Pure and Appl. Phys., 15, 228-233.
Roe, R., Bacchetta, V. L., Wong, P. M. G. 1967. J. of Phys. Chem., 71, 4190.
Rotenberg, Y., Boruvka, L., and Neumann, A. W. 1981. J. of Colloid and Inter. Sci., 93, 1, 169.
Scarborough, J. B. 1966. Numerical Mathematical Analysis, John Hopkins Press.
Scheflans, L., and Jacobs, M. B. 1953. The Handbook of Solvents, D. Van Nostrand Company, Inc., New York.
Scott, T. C. 1986. Chemical Separations, I, edited by C. Judson King and J. D. Navratil, Litarvan Literature, Denver, 313.
Stauffer, C. E. 1965. J. of Phys. Chem., 69, 1933.
Timmermans, J. 1959. Physio-Chemical Constants of Binary Systems, 2, Inter science Publishers, Inc., New York.
Vos, H., and Los, J. M. 1980. J. Colloid and Inter. Sci., 74, 2, 360.
Wenworth, W. E. 1965. J. Chem. Ed., 42, 96.
Wham, R. M., and Byers, C. H. 1985. "Mass Transfer from Single Droplets in Imposed Electric Fields," Presented at the Fourth symposium on Separation Science and Technology for Energy Applications, Knoxville, Tennessee, October 20-24.
Whittman, D. 1982. Program NONLIN, Version 3.0 Public Domain Software for IBM Personal Computer.
Winkel, D. 1965. J. of Phys. Chem., 69, 6.

APPENDIX A

DENSITY DATA

Table A.1. Binary liquid density[a] Data for interfacial tension calculations

Liquid	Temperature ($^\circ C$)						
Mixture	19.4	22.3	25.0	27.1	30.5	40.5	41.3
W-s-2EH	0.9970		0.9957		0.9940		0.9896
2EH-s-W	0.8354		0.8312		0.8272		0.8186
EG-s-CH	——	1.1092	1.1063	1.1056	1.1023	1.0961	1.0947
CH-s-EG	——	0.7706	0.7699	0.7692	0.7669	0.7570	0.7557
G-s-CH		1.2486		1.2456		1.2376	
CH-s-G		0.7744		0.7700		0.7574	

[a] density values were obtained by pycnometers and a Mettler balance

W = water

2EH = 2-ethyl hexanol

EG = ethylene glycol

CH = cyclohexane

G = glycerol

a-s-b = liquid phase a saturated with component b.

Table A.2. Density data for water-salt-2-ethyl hexanol system (Temperature - 28.5 °C)

Conc. of Salt in H_2O (g/mL)	Density (g/mL)	
	Water phase	2–ethyl hexanol phase
0.3165	1.2006	0.8315
0.1700	1.1007	0.8315
0.1001	1.0462	0.8315

APPENDIX B

EXAMPLE OF INPUT–OUTPUT DATA SETS

TYPICAL INPUT FILE
(Input file: gc201)

The following data in the left-most column represents the information that is required in the nonlinear parameter estimation computer programs. The right most column list the designation of each input. This data file is specifically for the FORTRAN 77 program PENDR.F. To use this data set in the Basic computer programs, NONLI4.BAS and NONLI8.BAS, the parameter names must be put in quotation marks, i.e., "r0".

153	No. of Data Points
7	No. of Iterations
0	A flag for weighting factor
0	A flag for weighting factor
1.00	fraction of calculated change to apply to each parameter
0.00	y coordinate
0.00	x coordinate
0.02	y coordinate
14.70	x coordinate
0.02	y coordinate
-18.30	x coordinate
2.66	y coordinate
23.70	x coordinate
(same as above)	
r0	Parameter name
73.00	Initial parameter estimate
beta	Parameter name
-0.00006	Initial parameter estimate
x0	Parameter name
0.	Initial parameter estimate
y0	Parameter name
0.	Initial parameter estimate

TYPICAL OUTPUT FILE FROM PENDR.F

The following printout is a typical output file generated by PENDR.F. Values for the x-scaling factor, density difference are listed. Parameter estimates, and values for the objective function are given for each iteration. Calculated and experimental profile coordinates are listed for the last iterations. These coordinates are used to generate graphs of the data.

DATA FILE: gc201

scale, cm/division, = 1.5910000000000d-03

density difference, g/cm**3, = 0.47330000000000

```
iteration #    1
r0=      73.0009336      beta= -0.0000600
x0= 0.0007149      y0= -0.0000123
 surface/interfacial tension = 19.56
 sum(objective function)**2 = 0.119311e+05      delta sum=
-0.100000e+21

iteration # 2
r0= 80.2662652      beta= -0.0000480
x0= 0.1433751      y0= -0.0813675
 surface/interfacial tension = 24.44
 sum(objective function)**2 = 0.118446e+05      delta sum=
-0.864836e+02

iteration # 3
r0= 80.7848293      beta= -0.0000477
x0= 0.1828252      y0= -0.0289267
 surface/interfacial tension = 24.61
 sum(objective function)**2 = 0.178079e+03      delta sum=
-0.116665e+05

iteration # 4
r0= 80.7689866      beta= -0.0000477
x0= 0.1830654      y0= -0.0374999
 surface/interfacial tension = 24.59
 sum(objective function)**2 = 0.954635e+02      delta sum=
-0.826154e+02

iteration # 5
r0= 80.7893214      beta= -0.0000477
x0= 0.1825366      y0= -0.0259211
 surface/interfacial tension = 24.62
 sum(objective function)**2 = 0.950903e+02      delta sum=
-0.373249e+00
```

```
iteration # 6
r0= 80.7889723        beta= -0.0000477
x0= 0.1826099       y0= -0.0263197
 surface/interfacial tension = 24.62
 sum(objective function)**2 = 0.948292e+02       delta sum=
-0.261105e+00
```

iteration # 7

Function Values

xcalc	xobs	ycalc	yobs
0.1827370	-0.1826085	0.0009137	0.0263225
14.3003109	14.5173915	1.275577	0.0463225
18.0281135	-18.4826085	2.0361469	0.0463225
23.2935166	23.5173915	3.4257134	2.6863225
25.9504270	-26.4826085	4.2727075	2.6863225
30.3090837	30.5173915	5.8836493	5.3663225
31.9692839	-32.4826085	6.5722549	5.3663225
35.4609236	35.5173915	8.1628065	8.0463225
37.0271070	-37.4826085	8.9411028	8.0463225
40.4735199	40.5173915	10.8030401	10.7263225
41.1942788	-41.4826085	11.2194164	10.7263225
45.2781600	45.5173915	13.7659956	13.4063225
45.2540472	-45.4826085	13.7499575	13.4063225
48.5451397	48.5173915	16.0486383	16.0863225
49.1654666	-49.4826085	16.5084944	16.0863225
51.7294778	51.5173915	18.5052377	18.7663225
52.3076874	-52.4826085	18.9775417	18.7663225
54.8081010	54.5173915	21.1210207	21.4463225
55.3421200	-55.4826085	21.6005910	21.4463225
57.7607359	57.5173915	23.8783537	24.1263225
58.2492286	-58.4826085	24.3597556	24.1263225
60.5702142	60.5173915	26.7572530	26.8063225
60.5541050	-60.4826085	26.7399101	26.8063225
63.2226882	63.5173915	29.7359186	29.4863225
63.2081605	-63.4826085	29.7187657	29.4863225
65.3321136	65.5173915	32.3120248	32.1663225
65.3188232	-65.4826085	32.2951248	32.1663225
66.9959547	66.5173915	34.4904770	34.8463225
66.9835716	-66.4826085	34.4738234	34.8463225
68.9458054	68.5173915	37.2309562	37.5263225
68.9345999	-68.4826085	37.2147030	37.5263225
70.4983900	69.5173915	39.5800783	40.2063225
70.7790884	-70.4826085	40.0207119	40.2063225
72.2591914	71.5173915	42.4486399	42.8863225
72.2502101	-71.4826085	42.4334180	42.8863225
73.9024562	73.5173915	45.3568100	45.5663225
73.8945119	-73.4826085	45.3422089	45.5663225

Function Values (cont.)

xcalc	xobs	ycalc	yobs
75.2258342	74.5173915	47.8918823	48.2463225
75.2188714	-74.4826085	47.8779652	48.2463225
76.4769147	75.5173915	50.4765518	50.9263225
76.6494148	-76.4826085	50.8498490	50.9263225
77.6577418	76.5173915	53.1180288	53.6063225
77.8072941	-77.4826085	53.4672923	53.6063225
78.8944752	78.5173915	56.1395429	56.2863225
78.8898991	-78.4826085	56.1277862	56.2863225
79.8983513	79.5173915	58.8288012	58.9663225
79.8943418	-79.4826085	58.8176937	58.9663225
80.9173306	81.5173915	61.8410090	61.6463225
80.8187764	-80.4826085	61.5372508	61.6463225
81.7431028	82.5173915	64.5501873	64.3263225
81.5818634	-80.4826085	63.9989061	64.3263225
82.4273997	82.5173915	67.0300196	67.0063225
82.3602846	-81.4826085	66.7751767	67.0063225
83.1086604	83.5173915	69.7802805	69.6863225
83.0546739	-82.4826085	69.5500755	69.6863225
83.6694387	83.5173915	72.3350903	72.3663225
83.6275525	-82.4826085	72.1311576	72.3663225
84.2025648	84.5173915	75.1034250	75.0463225
84.1696038	-83.4826085	74.9217084	75.0463225
84.6343727	84.5173915	77.7087629	77.7263225
84.6098378	-83.4826085	77.5484026	77.7263225
85.0037465	84.5173915	80.3445071	80.4063225
84.9872977	-83.4826085	80.2150839	80.4063225
85.3209740	85.5173915	83.1068446	83.0863225
85.3097969	-84.4826085	82.9998822	83.0863225
85.5588311	85.5173915	85.7632145	85.7663225
85.5530531	-84.4826085	85.6861076	85.7663225
85.7330363	85.5173915	88.4348561	88.4463225
85.7301202	-84.4826085	88.3800020	88.4463225
85.8443597	85.5173915	91.1159527	91.1263225
85.8433204	-84.4826085	91.0831761	91.1263225
85.8925115	86.5173915	93.8085511	93.8063225
85.8924983	-85.4826085	93.8048563	93.8063225
85.8779402	86.5173915	96.4753214	96.4863225
85.8776341	-85.4826085	96.4931142	96.4863225
85.8034011	86.5173915	99.1394017	99.1663225
85.8019324	-85.4826085	99.1783587	99.1663225
85.6661315	85.5173915	101.8559341	101.8463225
85.6659868	-85.4826085	101.8581729	101.8463225

Function Values (cont.)

xcalc	xobs	ycalc	yobs
85.4711456	85.5173915	104.5224098	104.5263225
85.4638002	-84.4826085	104.6092827	104.5263225
85.2197902	85.5173915	107.1753001	107.2063225
85.2086686	-84.4826085	107.2819993	107.2063225
84.9138712	85.5173915	109.8116744	109.8863225
84.8971809	-84.4826085	109.9402627	109.8863225
84.5549262	85.5173915	112.4226532	112.5663225
84.5323724	-84.4826085	112.5737479	112.5663225
84.1176388	84.5173915	115.1790806	115.2463225
84.0891681	-83.4826085	115.3483446	115.2463225
83.6535040	84.5173915	117.764844	1117.9263225
83.6174849	-83.4826085	117.9523653	117.9263225
83.1010057	83.5173915	120.5182409	120.6063225
83.0995161	-83.4826085	120.5252830	120.6063225
82.5391339	83.5173915	123.0615777	123.2863225
82.4872569	-82.4826085	123.2873874	123.2863225
81.9355371	83.5173915	125.5744654	125.9663225
81.8153793	-81.4826085	126.0507308	125.9663225
81.2272254	82.5173915	128.2962517	128.6463225
81.0867784	-80.4826085	128.8137801	128.6463225
80.4626615	81.5173915	131.0217447	131.3263225
80.3823451	-80.4826085	131.2967891	131.3263225
79.6463510	80.5173915	133.7348789	134.0063225
79.5547559	-79.4826085	134.0288026	134.0063225
78.7780017	79.5173915	136.4434569	136.6863225
78.6758624	-78.4826085	136.7508691	136.6863225
77.8614727	78.5173915	139.1364376	139.3663225
77.8576679	-78.4826085	139.1472936	139.3663225
76.8976270	77.5173915	141.8190955	142.0463225
76.8935059	-77.4826085	141.8303361	142.0463225
75.8898593	76.5173915	144.4830469	144.7263225
75.8853152	-76.4826085	144.4947684	144.7263225
74.6981638	74.5173915	147.4800889	147.4063225
74.8347859	-75.4826085	147.1452245	147.4063225
73.5986210	73.5173915	150.1206748	150.0863225
73.5933439	-73.4826085	150.1331524	150.0863225
72.4613873	72.5173915	152.7415624	152.7663225
72.4557014	-72.4826085	152.7544242	152.7663225
68.8492121	-69.4826085	160.4958321	160.8063225

Function Values (cont.)

xcalc	xobs	ycalc	yobs
71.2907465	71.5173915	155.3419737	155.4463225
71.2846636	-71.4826085	155.3551859	155.4463225
70.0873921	70.5173915	157.9226977	158.1263225
70.0810215	-70.4826085	157.9361508	158.1263225
68.6622133	68.5173915	160.8773113	160.8063225
67.3948306	67.5173915	163.4243360	163.4863225
67.3877465	-67.4826085	163.4383432	163.4863225
65.8915129	65.5173915	166.3608917	166.1663225
66.0958440	-66.4826085	165.9665113	166.1663225
64.3505980	63.5173915	169.2904618	168.8463225
64.7831449	-65.4826085	168.4756445	168.8463225
63.0060057	62.5173915	171.7910557	171.5263225
63.4502911	-64.4826085	170.9699142	171.5263225
61.6452835	61.5173915	174.2769249	174.2063225
62.1018570	-63.4826085	173.4473779	174.2063225
60.0330276	59.5173915	177.1753545	176.8863225
60.5013078	-61.4826085	176.3381694	176.8863225
58.4017370	57.5173915	180.0678025	179.5663225
59.1203369	-60.4826085	178.7979204	179.5663225
56.7573353	55.5173915	182.9549001	182.2463225
57.4858675	-58.4826085	181.6786339	182.2463225
55.3536755	54.5173915	185.4059598	184.9263225
55.8397435	-56.4826085	184.5580857	184.9263225
53.7022210	52.5173915	188.2853452	187.6063225
54.4359256	-55.4826085	187.0058960	187.6063225
52.3013058	51.5173915	190.7339405	190.2863225
52.7849708	-53.4826085	189.8873556	190.2863225
50.6643379	49.5173915	193.6151594	192.9663225
51.1412883	-51.4826085	192.7726844	192.9663225
49.0468715	47.5173915	196.4988247	195.6463225
49.5129306	-49.4826085	195.6632921	195.6463225
47.2334954	44.5173915	199.7996690	198.3263225
48.1403501	-48.4826085	198.1384047	198.3263225
46.7849430	-47.4826085	200.6303920	201.0063225

r0= 80.7889700 beta= -0.0000477
x0=0.1826085 y0= -0.02632- 25
surface/interfacial tension =24.62
sum(objective function)**2 =0.950652e+02 delta sum= 0.236073e+00

APPENDIX C

COMPUTER PROGRAMS

AND

THE RUNGE-KUTTA METHOD FOR SIMULTANEOUS FIRST ORDER DIFFERENTIAL EQUATIONS

PFIND4.BAS: A SONY BASIC COMPUTER PROGRAM
(Written for the Sony SMI-70 Graphics Computer)

This program is used to define drop edge coordinates of a pendant drop picture that has been manually traced with the aid of a Sony Graphics program. The interior of the pendant drop picture is filled with a particular color code which is different from the color code used to trace the perimeter of the drop. PFIND4.BAS scans the Graphics screen at each y location to determine the boundary at which the two colors intersect. This program stores the x,y drop edge coordinates, calculates the initial estimate of the x coordinate at the origin, xo, and prints the initial estimate of yo.

```
5 CLOSE
10 DIM A(300,2),NG$(9),VA(9,4)
15 CCLEAR 0,24
17 INPUT "READ DATA FROM DISK";K$: Rem ENTER N
18 IF K$<>"N" AND K$<>"Y" THEN 17
19 IF K$="Y" THEN 505
```

Insert User Defined Parmeters

```
20 INPUT "ENTER NO. OF GRAPHS";NG
21 PRINT ,"NAME ","XSCALE","YSCALE", "DENSITY"
22 FOR I=1 TO NG
23 INPUT "ENTER NAMES OF GRAPHS",NG$(I)
24 VA(I,1)=VA(I-1,1):VA(I,2)=VA(I-1,2):IF SS$="S" THEN 27
25 INPUT "ENTER X,Y SCALE";VA(I,1),VA(I,2)
26 INPUT "ENTER S IF SCALE IS THE SAME FOR ALL GRAPHS"; SS$
27 INPUT "ENTER DENSITY DIFF (G/ML)";VA(I,3)
29 INPUT "ENTER Y-INCREMENT";VA(I,4)
30 PRINT ,NG$(I),VA(I,1),VA(I,2),VA(I,3),VA(I,4)
31 NEXT I
40 CCLEAR .24
60 FOR JP=1 TO NG
61 BG$="A:"+NG$(JP):XS=VA(JP,1):YS=VA(JP,2):DD= VA(JP,3) :
 K=VA(JP,4)
```

Load Pendant Drop Picture

```
70 _ GLOAD (BG$)
```

Vertical Scan of Graphics Screen to
Determine Drop Edge Coordinates

```
85 N=0:L=0:P=319
90 FOR I=0 TO 199 STEP K
100 M=0
106 IF GPOINT(L,I)<>10 THEN 110:L=0
```

```
107 IF GPOINT(P,I)<>10 THEN 110:P=319
110 FOR J=L TO 319
120 IF M=1 THEN 160
130 IF GPOINT(J,I)<>10 THEN 200
140 N=N+1:M=M+1:A(N,1)=J-1:A(N,2)=I
150 GO TO 200
160 IF GPOINT(P,I)<>10 THEN P=P-1
161 IF N-2<1 THEN 165
165 IF GPOINT(P,I)<>10 THEN 160
170 N=N+1:A(N,1)=P+1:A(N,2)=I
180 GO TO 205
200 NEXT J
205 IF N<4 THEN 260
210 IF M<>1 AND N¿0 THEN 270
215 L=A(N-1,1)-10:P=A(N,1)+10
217 IF N-3<1 THEN 260
220 IF A(N-1,1)>A(N-3,1) THEN L=A(N-1,1)
230 IF (A(N,1))<A(N-2,1) THEN P=A(N,1)
260 NEXT I
```

Clear Screen and Plot x,y Drop Edge Coordinates

```
270 GCLEAR 9
380 FOR I=1 TO N
390 GPLOT(A(I,1),A(I,2)),10,0
400 NEXT I
```

Save Drop Edge Coordinates

```
425 FS$="B:"+NG$(JP)
430 OPEN /S FS$
432 N1$=STR$(N)+","
435 SEND N1$
440 FOR I=1 TO N
445 C$=STR$(A(I,1))+","
450 SEND C$
455 D$=STR$(A(I,2))+","
460 SEND D$
465 NEXT I
470 CLOSE
480 GOTO 575
500 GOTO 575
```

Receive Drop Edge Coordinate data from Existing File
(This was not required for the present research)

```
505 INPUT "ENTER DRIVE#-FILENAME";FR$
510 OPEN /R FR$
515 RECEIVE N1$:N=VAL(N1$)
```

```
520 FOR I=1 TO N
530 RECEIVE C$:A(I,1)=VAL(C$)
540 RECEIVE D$:A(I,2)=VAL(D$)
550 NEXT I
560 CLOSE
```

Calculate xo and Print x,y Drop Edge Coordinates

```
565 REM GOTO 610
575 PRINT , :PRINT ,NG$(JP):PRINT ," "
576 REM GOTO 610
577 PRINT , : PRINT ,"X","Y"
590 PRINT ,A(N,1),A(N,2)
611 SUM2=0
612 FOR I=1 TO N-1 STEP 2:SUM2=A(I+1,1)+A(I,1)+SUM2
613 NEXT I
614 X0=SUM2/N:PRINT ,"XX0= ",X0:SUM2=0:GCLEAR 0:CCLEAR 1,24
615 NEXT JP
620 CLOSE
700 END
```

RK4TR8.BAS: A MICROSOFT BASIC COMPUTER PROGRAM
(Written for the IBM Personal Computer)

This program converts drop edge data files that were transfered from the SONY SMI-70 Graphics Computer into a data file with the proper format for use in the nonlinear parameter estimation programs, NONLI4.BAS, and NONLI8.BAS. To use this data file in the FORTRAN 77 program PENDR.F, the quotation marks

around the parameter names must be removed.

The user must insert the following data:

1. rxy - The x-y scaling factor x divisions per cm / y divisions per cm.
2. Input file name
3. Output file name
4. Estimates of Xo and Yo, Sony Graphic divisions.

RK4TR8.BAS will then input the data file, correct the data set about the origin, sort the data in ascending Y coordinates, determine initial estimates for Ro, Beta, Xo and Yo, and will finally store the data in the proper format.

```
4 DIM XG(4,4),YG(4),SG(4),LG(4)
5 REM INPUT X-Y SCALING FACTOR (X DIVISIONS PER CM/ Y DIVISIONS
PER CM)
7 INPUT "enter rxy";RXY
10 DIM VOBS(450,1),FOBS(450),VOB(450),FOB(450)
12 INPUT "enter file name";FG$
13 IF FG$="" THEN 12
14 OPEN "i",#2,FG$
15 INPUT "ENTER OUTPUT FILE NAME";OP$: OPEN OP$ FOR OUTPUT AS #1
16 INPUT #2,N1$:N=LEN(N1$):N1$=RIGHT$(N1$,N-1):N1=VAL(N1$)
17 N1=N1:GOSUB 3000:REM INPUT SONY DATA FILE
20 GOSUB 4000: REM ESTIMATE POSITION OF ORIGIN BY CURVEFITTING
   THE BOTTOM OF THEPENDANT DROP TO A QUADRATIC EXPRESSION AND
   DEFINING THE ORIGIN AS THE MINIMUM OF THE FUNCTION.
25 INPUT "enter xx0,yy0";XX0,YY0: REM USER INPUTS ESTIMATES OF
   XX0 AND YY0.
30 FOR I=1 TO N1
40 VOBS(I,1)=VOBS(I,1)-XX0
50 FOBS(I)=ABS(FOBS(I)-YY0)
60 NEXT I
99 REM USER DEFINES HOW MUCH OF THE DROP PROFILE TO CURVEFIT
100 INPUT"enter left(l), right(r) or both(b) sides";S$
102 INPUT"enter s2-increment";S2
103 J=0
105 IF S$="l" THEN S1=1:S2=-S2
110 IF S$="r" THEN S1=N1
111 IF S$<>"b" THEN 122
112 B1=1
```

```
115 IF S$="b" AND B1=1 THEN S1=1:S2=-S2
120 IF S$="b" AND B1=2 THEN S1=N1:S2=-S2
122 B1=B1+1:SS=0
130 FOR I= N1/2 TO S1 STEP S2
132 J=J+1
133 IF I+S2<0 THEN 170
135 REM LPRINT USING "####.#### ";VOBS(I+1,1),VOBS(I-1,1),
    FOBS(I+1),FOBS(I-1)
136 IF ABS(VOBS(I+S2,1)+VOBS(I,1))<ABS(VOBS(I+S2,1))+
    ABS(VOBS(I,1)) THEN 170
137 IF SS=0 THEN SS=(VOBS(I,1)^ 2+FOBS(I)^ 2)^ .5
140 SS=((VOBS(I+S2,1)-VOBS((I),1))^ 2+
    (FOBS(I+S2)-FOBS(I))^ 2)^ .5+SS
150 FOB(J)=FOBS(I)
155 VOB(J)=VOBS(I,1)
160 PRINT VOB(J),FOB(J)
161 IF S$="b" GOTO 170
170 NEXT I
171 IF S$="b" AND B1<3 THEN 120
173 GOSUB 5000: REM SORT DATA IN ACCENDING Y-ORDER AND STORE
    CORRECTED DATA FILE.
175 CLOSE #1: CLOSE #2
180 END
3000 FOR I=1 TO N1/2
3010 I1=I
3020 FOR L=1 TO 2
3030 IF L=2 THEN I1=N1-I+1
3040 FOR K=1 TO 2
3050 INPUT #2,V$
3052 N=LEN(V$):V$=RIGHT$(V$,N-1)
3060 IF K=1 THEN VOBS(I1,1)=VAL(V$)-XX0
3070 IF K>1 THEN FOBS(I1)=ABS(RXY*VAL(V$))
3090 NEXT K
3100 NEXT L
3110 NEXT I
3120 RETURN
4000 FOR IG=1 TO 3
4010 FOR JG=1 TO 3
4020 XG(JG,KG)=0
4030 NEXT JG
4040 NEXT IG
4050 YG(1)=0
4060 YG(2)=0
4070 YG(3)=0
4080 FOR IG=N1/2-5 TO N1/2+4
4085 PRINT VOBS(IG,1),FOBS(IG)
4090 XG(1,1)=VOBS(IG,1)^ 4 + XG(1,1)
```

```
4100 XG(1,2)=VOBS(IG,1)^ 3 + XG(1,2)
4110 XG(1,3)=VOBS(IG,1)^ 2 + XG(1,3)
4120 XG(2,1)=XG(1,2)
4130 XG(2,2)=XG(1,3)
4140 XG(2,3)=VOBS(IG,1) + XG(2,3)
4150 XG(3,1)=XG(1,3)
4160 XG(3,2)=XG(2,3)
4170 XG(3,3)=XG(3,3)+1
4180 YG(1)=FOBS(IG)*VOBS(IG,1)^ 2 + YG(1)
4190 YG(2)=FOBS(IG)*VOBS(IG,1) + YG(2)
4200 YG(3)=FOBS(IG) + YG(3)
4210 NEXT IG
4220 FOR IG= 1 TO 3
4225 LG(IG)=IG
4230 SG(IG)=0
4240 FOR JG=1 TO 3
4250 IF SG(IG)<ABS(XG(IG,JG)) THEN SG(IG)=ABS(XG(IG,JG))
4260 NEXT JG
4280 NEXT IG
4290 FOR KG= 1 TO 2
4300 RMAX=0
4310 FOR IG= KG TO 3
4320 R=ABS(XG(LG(IG),KG))/SG(LG(IG))
4330 IF R<RMAX GOTO 4360
4340 JG=IG
4350 RMAX=R
4360 NEXT IG
4370 LK=LG(JG)
4380 LG(JG)=LG(KG)
4390 LG(KG)=LK
4400 FOR IG=KG+1 TO 3
4410 XMULT=XG(LG(IG),KG)/XG(LK,KG)
4420 XG(LG(IG),KG)=XMULT
4430 FOR JG=KG+1 TO 3
4440 XG(LG(IG),JG)=XG(LG(IG),JG)-XMULT*XG(LK,JG)
4450 NEXT JG
4460 NEXT IG
4470 NEXT KG
4480 FOR JG=1 TO 2
4490 FOR IG=JG+1 TO 3
4500 YG(LG(IG))=YG(LG(IG))-XG(LG(IG),JG)*YG(LG(JG))
4505 NEXT IG
4507 NEXT JG
4510 SG(3)=YG(LG(3))/XG(LG(3),3)
4520 FOR IG=1 TO 2
4530 SUMGG=YG(LG(3-IG))
4540 FOR JG=3-IG+1 TO 3
```

```
4550 SUMGG=SUMGG-XG(LG(3-IG),JG)*SG(JG)
4560 NEXT JG
4570 SG(3-IG)=SUMGG/XG(LG(3-IG),3-IG)
4580 NEXT IG
4590 XX0=-SG(2)/(2*SG(1))
4600 YY0=SG(1)*XX0^ 2+SG(2)*XX0+SG(3)
4605 PRINT "xx0= ";XX0;" yy0= ";YY0
4610 RETURN
5000 N1=J
5001 XE=0:WRITE #1,N1: REM STORE NUMBER OF DATA POINTS.
5002 WRITE #1,10: REM STORE NUMBER OF ITERATIONS.
5003 WRITE #1,0:WRITE #1,0: REM STORE NUMBER WEIGHTING FACTORS.
5004 WRITE #1,1: REM FRACTION OF THE CALCULATED CHANGE TO APPLY
     TO EACH PARAMETER.
5005 FOR I=1 TO N1
5010 FOR J=I+1 TO N1
5020 IF FOB(I)<FOB(J) GOTO 5050
5030 DV=VOB(I):VOB(I)=VOB(J):VOB(J)=DV
5040 EV=FOB(I):FOB(I)=FOB(J):FOB(J)=EV
5050 NEXT J
5055 IF ABS(VOB(I))>(XE/.95) THEN XE=.95*ABS(VOB(I)): REM
     ESTIMATE RO
5060 WRITE #1,FOB(I): WRITE #1,VOB(I): REM STORE CORRECTED
     PROFILE DATA POINTS.
5080 NEXT I
5081 BETA = .3/(XE^ 2): REM ESTIMATE BETA.
5082 WRITE #1,"RO":WRITE #1,XE: REM STORE INITIAL ESTIMATE OF RO.
5084 WRITE #1,"BETA":WRITE #1,BETA: REM STORE INITIAL ESTIMATE OF
     BETA.
5086 WRITE #1,"XXO":WRITE #1,0: REM STORE INITIAL ESTIMATES OF
     X0.
5088 WRITE #1,"YY0' ':WRITE #1,0: REM STORE INITIAL ESTIMATES OF
     Y0.
5090 RETURN
```

NONLI4.BAS: A MICROSOFT BASIC COMPUTER PROGRAM
(Written for the IBM Personal Computer)

This program uses nonlinear parameter estimation algorithms to curvefit experimental pendant drop profile data to the Laplace equation. The core of the program consists of algorithms that were written by Dave Whitman for solving nonlinear parameter problems. The program was modified and adapted by Michael T. Harris for curvefitting axisymmetric drop edge data to the Laplace equation. Numerical derivatives are used to form the b-matrix.

The user must insert the following data:

1. Number of data files to be processed.
2. Full name of data file.
3. The x-scaling factor, i.e. cm/(measured unit).
4. Density difference between the saturated phases, g/cm^3.

A listing of the program is given below. Lines 11000-11900 contains the fourth-order Runge-Kutta method for solving a set of first order differential equations to determine xcalc and ycalc. This portion of the program was developed by Michael T. Harris (1986).

```
15 DEFINT I-N
16 DEFDBL A-H,O-Z
110 DIM AOBS(300),FCALC(300),FTEMP(300),OBSWT(300)
120 DIM VOBS(300,2),V(300,2),VARWT(300,2),DFDV(300,2)
130 DIM P(4),PNAME$(4), DFDP(300,4),INDEX(4,5),B(4,4)
140 DIM IFLAG(300),DLAMBDA(300),RHS(4),BCD(4,4),FCALC2(300)
150 DIM IFIL1$(30),XSCA1(30),DENDI1(30)
10050 NVAR=1 : NP=2
10052 NFT=3
10055 IYT=1:NONDIV=0:NONCONVERGE=0
10060 GOSUB 18000 'initialization routine
10065 IFT=IFT+1
10080 FOR IT = 1 TO NUMIT
10090 'print progress report on screen
10100 GOSUB 12000 'subroutine iteration report
10110 'Test for non-convergance, exit if so
10120 GOSUB 13000 'subroutine converge
10130 IF NONCONVERGE = 1
 THEN PRINT"nonconvergence termination" :
 GOTO 10270
10135 IF NONDIV=1 THEN PRINT "convergence": GOTO 10270
10140 'Calculate lagrangian multipliers
10150 GOSUB 13500 'subroutine lambda
10160 'If internal weighting desired, calculate new obswts
10170 IF INTERNALWT = 1
 THEN GOSUB 14500
10180 'Set up matrix equation to get parameter changes
```

```
10190 GOSUB 15000 'SUBROUTINE SETUP
10200 'Solve equation for parameter changes
10210 GOSUB 16000 'subroutine solve
10220 'Apply changes
10230 GOSUB 17000 'subroutine deltap
10240 NEXT IT
10250 '
10260 'print final report
10270 GOSUB 19000 'subroutine report
10280 'Do any final processing (user supplied)
10290 GOSUB 20000
10295 IF IFT<=NFT THEN GOTO 10055
10300 END
11000 DS=.006579*P(1)
11010 S=0:JCOUNT=0:IERR=1
11020 THI1=0
11030 THI=THI1
11040 XCALC=0
11050 YCALC=0
11070 REM runge-kutta method
11080 J=J
11085 XOLD=XCALC
11090 YOLD=YCALC
11110 THOLD=THI
11120 SP=S/P(1)
11125 SF2=SP:THF2=THOLD
11130 GOSUB 11740
11135 XK1=VF2
11140 XK1=XK1*DS
11145 SF1=SP:THF1=THOLD
11150 GOSUB 11710
11155 XL1=VF1
11160 XL1=XL1*DS
11165 SF1=SP:XF3=XCALC:YF3=YCALC:THF1=THOLD
11170 GOSUB 11770
11175 XM1=VF3
11180 XM1=XM1*DS
11190 SP=(S+DS/2!)/P(1)
11200 XP=XCALC+XK1/2!
11220 YP=YCALC+XL1/2!
11230 TIP=THOLD+XM1/2!
11235 SF2=SP:THF2=TIP
11240 GOSUB 11740
11245 XK2=VF2
11250 XK2=XK2*DS
11253 SF1=SP:THF1=TIP
11255 GOSUB 11710
```

```
11256 XL2=VF1
11257 XL2=XL2*DS
11258 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11260 GOSUB 11770
11265 XM2=VF3
11270 XM2=XM2*DS
11280 XP=XCALC +XK2/2!
11290 YP= YCALC+XL2/2
11310 TIP=THI+XM2/2!
11315 SF2=SP:THF2=TIP
11320 GOSUB 11740
11325 XK3=VF2
11330 XK3=XK3*DS
11335 SF1=SP:THF1=TIP
11340 GOSUB 11710
11345 XL3=VF1
11350 XL3=XL3*DS
11355 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11360 GOSUB 11770
11365 XM3=VF3
11370 XM3=XM3*DS
11380 SP=(S+DS)/P(1)
11390 XP=XCALC +XK3
11410 YP=YCALC + XL3
11420 TIP=THI +XM3
11425 SF2=SP:THF2=TIP
11430 GOSUB 11740
11435 XK4=VF2
11437 SF1=SP:THF1=TIP
11440 GOSUB 11710
11445 XL4=VF1
11447 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11450 GOSUB 11770
11460 S=S+DS
11465 XM4=VF3
11470 XCALC=XCALC+(XK1+2!*XK2+2!*XK3+XK4*DS)/6!
11475 YCALC=YCALC+(XL1+2!*XL2+2!*XL3+XL4*DS)/6!
11477 REM YCALC=YCALC+(-(XCALC-XOLD)^ 2+DS^ 2)^ .5
11490 THI=THI +(XM1+2!*XM2+2!*XM3+XM4*DS)/6!
11492 IF THI>3.14 OR THI<0 GOTO 11580
11493 IF IMIN<=1 GOTO 11500
11500 IMIN=IERR
11505 FOR I=IMIN TO NOBS
11510 X0=XOLD:Y0=YOLD:X1=XCALC:Y1=YCALC:XS=ABS(VOBS(I,1)):
 YS=AOBS(I)
11512 REM IF DII=2 THEN XS=VOBS(I,1)+P(3):DII=1
11514 DII=DII+1
```

```
11515 DELT=(X1-X0)^ 2+(Y1-Y0)^ 2
11520 IF DELT=0 GOTO 11550
11522 STEST0=((X1-X0)*(XS-X0)+(Y1-Y0)*(YS-Y0))/DELT
11524 STEST1=((X1-X0)*(XS-X1)+(Y1-Y0)*(YS-Y1))/DELT
11526 IF STEST1<>0 GOTO 11532
11528 VOBS(I,2)=X1:FCALC(I)=Y1:FCALC2(I)=Y1
11529 FCALC(I)=((XS-VOBS(I,2))^ 2+(YS-FCALC(I))^ 2)^ .5
11530 JCOUNT=JCOUNT+1:IERR=I+1
11531 LOCATE 1,1:PRINT USING "### ";I,IERR:GOTO 11550
11532 IF STEST0=0 GOTO 11550
11534 STS=SGN(1!-STEST0):STS1=SGN(1!-STEST1)
11536 IF STS=STS1 GOTO 11550
11538 VOBS(I,2)=X0+STEST0*(X1-X0):FCALC(I)=Y0+STEST0 * (Y1-Y0):
    FCALC2(I)=FCALC(I)
11539 FCALC(I)=((XS-VOBS(I,2))^ 2+(YS-FCALC(I))^ 2)^ .5:IERR=I+1
11540 JCOUNT=JCOUNT+1
11550 LOCATE 1,1:PRINT USING "### ";I,IERR:NEXT I
11560 IF IERR<=NOBS GOTO 11080
11580 WEIGHT=(NOBS-2)/(JCOUNT-2)
11700 GOTO 11900
11710 REM sunb f1
11720 VF1=SIN(THF1)
11730 RETURN
11740 REM sun f2.f
11750 VF2=COS(THF2)
11760 RETURN
11770 REM f3.f
11780 GOSUB 11710
11785 ZF3=VF1
11787 IF XF3=0 THEN VF3=1/P(1):GOTO 11800
11790 VF3=2/P(1)+P(2)*YF3-ZF3/XF3
11800 RETURN
11900 RETURN
12010 '* SUBROUTINE ITERTATION REPORT
 *
12030 '* Prints out current parameters, function values, and
deviation *
12050 GOSUB 12200 'print parameters
12060 GOSUB 11000 'get function values in fcalc
12070 GOSUB 12500 'print function values and
deviation
12080 RETURN
12210 '* SUBROUTINE PARAMREPORT( IT, NP, P)
12220 '* 2/7/82 by Dave Whitman
12230 '* Prints out current parameter values
12250 CLS : LOCATE 4,4
12260 PRINT "Parameters, Iteration";IT : PRINT
```

```
12265 LPRINT "Parameters, Iteration";IT : LPRINT
12270 LOCATE 7,2
12280 COLOR 1 : PRINT " Name | Value | Change "; : COLOR 7 :
PRINT
12285 LPRINT " Name | Value | Change ";:LPRINT
12290 FOR I = 1 TO NP
12300 LOCATE I+7,2
12310 PRINT USING "    #.##### +####"; PNAME$(I);"|
"; P(I); " | ";-1 * RHS(I)
12311 SURT=-979.699*DENDIF*XSCAL^ 2/P(2):STFFD$="surface tension"
12312 IF I=2 THEN PRINT USING "    ###.####";
STFFD$;"| ";SURT
12315 LPRINT USING "    ###.####### +####.#########";
PNAME$(I);"| "; P(I); " | ";-1 * RHS(I)
12316 IF I=2 THEN LPRINT USING "    ###.####";
STFFD$;"|";SURT
12320 NEXT I
12330 PRINT
12335 LPRINT
12340 RETURN
12510 '* SUBROUTINE FUNCTIONREPORT
*
12520 '* Prints obs. and calc. function values, and deviation
between them *
12540 IROW = 1 : ICOL = 40 : IOFFSET = 20 : IROOM = 40 : NUMROWS
= 20
12550 LOCATE IROW,ICOL : COLOR 1
12560 PRINT " obs. | calc. ";:COLOR 7 : PRINT
12565 LPRINT " obs.| calc. ";:LPRINT
12570 IF NOBS >= NUMROWS
THEN LOCATE IROW,ICOL+IOFFSET : COLOR 1:
PRINT " obs. | calc. "; : COLOR 7 :
PRINT
12580 DEVSQ = 0
12590 FOR I = 1 TO NOBS
12600 IF I > IROOM THEN 12630
12610 IF I <= NUMROWS THEN LOCATE (IROW + I),ICOL
ELSE LOCATE (IROW + IMOD NUMROWS),(ICOL + IOFFSET)
12620 PRINT USING "#.####  #.####"; VOBS(I,1);" | ";
VOBS(I,2);
12630 DEV = FCALC(I)*WEIGHT
12640 DEVSQ = DEVSQ + DEV * DEV * OBSWT(I)
12650 NEXT I
12660 LOCATE 20,5 : PRINT USING "  ##.#####";"sum error^ 2 =
";DEVSQ
12665 LPRINT USING "  ##.#####"; "sum error^ 2 = ";DEVSQ
12670 IF IT = 1 THEN 12690 'no change in first iteration
```

```
12680 LOCATE 21,5: PRINT USING "  #.######";
 "Change = ";DEVSQ-DEVSQ1;
12685 LPRINT USING "   #.######";"Change = ";
 DEVSQ-DEVSQ1;
12690 RETURN
13010 '* SUBROUTINE CONVERGE ( ERRSQ, DEVSQ, DEVSQ1, DEVSQ2,
 NONCONVERGE )
13030 '* Compares squared deviation of calculated function from
 observed
13040 '* function with that obtained in the last 2 iterations. If
 the
13050 '* deviation got worse two iterations in a row, set
 nonconverge flag.
13070 IF (DEVSQ > 1.05*DEVSQ1 AND DEVSQ1 > 1.05 * DEVSQ2)
 THEN NONCONVERGE = 1
13075 IF ABS(DEVSQ-DEVSQ1)/DEVSQ<.0001 THEN NONDIV=1
13080 DEVSQ2 = DEVSQ1
13090 DEVSQ1 = DEVSQ
13100 RETURN
13510 '* SUBROUTINE LAMBDA ( DLAMBDA, FCALC, FOBS, OBSWT)
13530 '* Calculates lagrangian multipliers for setting up matrix
 equation
13550 FOR I = 1 TO NOBS
13560 DLAMBDA(I) = FCALC(I) * OBSWT(I)*WEIGHT
13570 NEXT I
13580 RETURN
14010 '* SUBROUTINE VSLOPE( V,DFDV,NVAR )
14030 '* Calculates the partials of the function w/ r.t. each
14040 '* of the variables at each of the observed points, and
14050 '* stores them in dfdv.
14070 GOSUB 11000 'call function( p, v, nobs, nvar, np, fcalc)
14080 FOR IW = 1 TO NOBS
14090 FTEMP(IW) = FCALC(IW)
14100 NEXT IW
14110 FOR IW = 1 TO NVAR
14120 FOR JW = 1 TO NOBS
14130 IFLAG(JW) = 0
14140 IF ABS(VOBS(JW,IW)) < 1D-20
 THEN VOBS(JW,IW) = .0005# : IFLAG(JW) = 1
 ELSE VOBS(JW,IW) = VOBS(JW,IW) * 1.0005#
14150 PRINT "modified var:" ;VOBS(JW,IW)
14160 NEXT JW
14170 GOSUB 11000 'call function(fcalc)
14180 FOR JW = 1 TO NOBS
14190 IF IFLAG(JW) = 1
 THEN DFDV(JW,IW) = (FCALC(JW) - FTEMP(JW)) / .0005# :
 VOBS(JW,IW) = VOBS(JW,IW) -.0005#
```

```
14200 IF IFLAG(JW) <> 1
 THEN DFDV(JW,IW) = (FCALC(JW)-FTEMP(JW)) /
 (.0005# * VOBS(JW,IW)): VOBS(JW,IW) =
 VOBS(JW,IW) / 1.0005#
14210 PRINT"dfdv(";JW;IW;")="; DFDV(JW,IW)
14220 NEXT JW
14230 NEXT IW
14240 RETURN
14510 '* subroutine weigh[ p, nobs, nvar, v, dfdv, varwt, obswt ]
14530 '* calculates new weights for function values,
14540 '* using the follwing formula:
14550 '* obswt(i) = 1/ sum [(dfdv)^ 2 * (1/varwt(v))]
14560 '* note: obswt(i) = 1/L(i) in Wentworth article
14580 IF IT = 1 THEN RETURN 'skip weighting on first iteration
14590 GOSUB 14000 'subroutine vslope
14600 FOR IW = 1 TO NOBS
14610 SUM = 0#
14620 FOR JW = 1 TO NVAR
14630 SUM = SUM + DFDV(IW,JW)*DFDV(IW,JW)/VARWT(IW,JW)
14640 NEXT JW
14650 OBSWT(IW) = 1# / SUM
14660 NEXT IW
14670 PRINT "new function weights:"
14680 FOR IW = 1 TO NOBS
14690 PRINT OBSWT(IW)
14700 NEXT IW
14710 RETURN
15000 '*******************************************************
15010 '* SUBROUTINE SETUP( B,RHS,dfdp) *
15020 '* Sets up matrix equation to get changes to parameters *
15030 '*******************************************************
15040 '
15050 'Get partials of function w.r.t. parameters
15060 GOSUB 17500 'subroutine pslope
15070 'Now set up matrices
15080 FOR I = 1 TO NP
15090 'Set up right hand side element
15100 RHS(I) = 0#
15110 FOR J = 1 TO NOBS
15120 RHS(I) = RHS(I) + DFDP(J,I) * DLAMBDA(J)
15130 NEXT J
15140 'Set up left hand side elements
15150 FOR J = 1 TO NP
15160 B(I,J) = 0#
15170 FOR K = 1 TO NOBS
15180 B(I,J) = B(I,J) + DFDP(K,I) * DFDP(K,J) *
 OBSWT(K)
```

```
15190 NEXT K
15200 NEXT J
15210 NEXT I
15220 RETURN
16000 '*****************************************************
16010 '* subroutine solve[b#(np,np), rhs(np), np] *
16020 '* 1/31/82 by Dave Whitman *
16030 '* solves matrix equations of the form b# x = rhs# *
16040 '* inverts b# in place,multiplies rhs# by inverse *
16050 '* uses Gauss-Jordan matrix inversion *
16060 '* for good results b# and rhs# must be dbl precision*
16070 '* ref: J.M. McCormick "Numerical Methods in FORTRAN"*
16080 '*****************************************************
16090 DETERM = 1#
16100 FOR I = 1 TO NP
16110 INDEX(I,3) = 0
16120 NEXT I
16130 FOR I = 1 TO NP 'MAIN LOOP
16140 'search for pivot element
16150 MAX# = 0#
16160 FOR J = 1 TO NP
16170 IF INDEX(J,3) = 1 THEN 16260
16180 FOR K = 1 TO NP
16190 IF INDEX(K,3) > 1 THEN 16700
16200 IF INDEX(K,3) = 1 THEN 16250
16210 IF MAX# > ABS(B(J,K)) THEN 16250
16220 IROW = J
16230 ICOL = K
16240 MAX# = ABS(B(J,K))
16250 NEXT K
16260 NEXT J
16270 INDEX(ICOL,3) = INDEX(ICOL,3) + 1
16280 INDEX(I,1) = IROW
16290 INDEX(I,2) = ICOL
16300 'interchange rows to put pivot on diagonal
16310 IF IROW = ICOL THEN 16380 'ALREADY THERE
16320 DETERM = -1# * DETERM
16330 FOR J = 1 TO NP
16335 PRINT "brow= ";B(IROW,J);" bcol= ";B(ICOL,J)
16340 SWAP B(IROW,J),B(ICOL,J)
16345 PRINT "brow= ";B(IROW,J);" bcol= ";B(ICOL,J)
16350 NEXT J
16355 PRINT "rrow= ";RHS(IROW);" rcol= ";RHS(ICOL)
16360 SWAP RHS(IROW),RHS(ICOL)
16370 'divide pivot row by pivot element
16380 PIVOT = B(ICOL,ICOL)
16390 DETERM = DETERM * PIVOT
```

```
16400 B(ICOL,ICOL) = 1#
16410 FOR J = 1 TO NP
16420 B(ICOL,J) = B(ICOL,J)/PIVOT
16430 NEXT J
16440 RHS(ICOL) = RHS(ICOL)/PIVOT
16450 ' reduce non-pivot rows
16460 FOR J = 1 TO NP
16470 IF J = ICOL THEN 16540
16480 T = B(J,ICOL)
16490 B(J,ICOL) = 0# :BCD(J,ICOL)=B(J,ICOL)
16500 FOR K = 1 TO NP
16510 B(J,K) = B(J,K) - B(ICOL,K)*T:BCD(J,K)=B(J,K)
16520 NEXT K
16530 RHS(J) = RHS(J) - RHS(ICOL)*T
16540 NEXT J
16550 NEXT I
16560 'interchange columns
16570 FOR I = NP TO 1 STEP -1
16580 IF INDEX(I,1) = INDEX(I,2) THEN 16640
16590 IROW = INDEX(I,1)
16600 ICOL = INDEX(I,2)
16610 FOR J = 1 TO NP
16620 SWAP B(J,IROW), B(J,ICOL)
16630 NEXT J
16640 NEXT I
16650 'test for singularity
16660 FOR I = 1 TO NP
16670 IF INDEX(I,3) <> 1 THEN 16700
16680 NEXT I
16690 RETURN
16700 PRINT"singular matrix error ":RETURN
17000 '*************************************************
17010 '* SUBROUTINE DELTAP ( P, RHS, NP ) *
17020 '* *
17030 '* Modifies parameters according to changes in rhs *
17040 '*************************************************
17050 FOR I = 1 TO NP
17060 P(I) = P(I) - RHS(I) * FRACT
17070 NEXT I
17080 RETURN
17510 '* subroutine pslope[ p, vobs, nobs, np, nvar,
 dfdp(nobs,np) ]
17520 '* 2/1/82 by Dave Whitman
 *
17530 '* calculates partial of the function with
 *
17540 '* respect to the each of the parameters at
```

```
*
17550 '* each of the observations, and stores them in dfdp.
*
17580 FOR IS = 1 TO NOBS
17590 FTEMP(IS) = FCALC(IS)*WEIGHT
17600 NEXT IS
17610 FOR IS = 1 TO NP
17620 TP = P(IS)
17625 P(IS) = TP*1.000000005#
17630 REM IF TP < 1D-20
 THEN P(IS) = TP + .000000005#
 ELSE P(IS) = TP * 1.000000005#
17640 GOSUB 11000 'call function( fcalc )
17645 DPP=0
17650 FOR JS = 1 TO NOBS
17660 REM IF TP < 1D-20 THEN DFDP(JS,IS) = (FCALC(JS)*WEIGHT -
 FTEMP(JS)) / .000000005# ELSE DFDP(JS,IS) =
 (FCALC(JS)*WEIGHT - FTEMP(JS)) / (.000000005# * TP)
17665 DFDP(JS,IS)=(FCALC(JS)*WEIGHT-FTEMP(JS))/(.000000005#*TP)
17670 NEXT JS
17675 REM FOR JS=1 TO NOBS
17676 REM DFDP(JS,IS)=(1-DPP/DEVSQ)/(ABS(1-DPP/DEVSQ))*
 DFDP(JS,IS)
17677 REM NEXT JS
17680 P(IS) = TP
17690 NEXT IS
17700 RETURN
18010 '* SUBROUTINE initialize
*
18020 '*
*
18030 '* Prompts for name of input file, then reads problem in
*
18040 '* from input file.
*
18060 '
18070 KEY OFF : CLS
18075 IF IFT >1 THEN 18086
18076 IFT=1: INPUT "enter # of files";NFT
18077 FOR I=1 TO NFT
18080 LOCATE 13,20 : INPUT "Name of input file? ",IFIL1$(I)
18082 INPUT "enter x-scale and density diff";XSCA1(I),DENDI1(I)
18084 NEXT I
18086 IFILE$=IFIL1$(IFT):XSCAL=XSCA1(IFT):DENDIF=DENDI1(IFT)
18090 OPEN IFILE$ FOR INPUT AS #1
18100 INPUT#1,NOBS
18150 INPUT#1,NUMIT
```

```
18160 INPUT#1,IUSERWT,INTERNALWT
18170 INPUT#1, FRACT 'fraction of calculated param. change to
 apply
18180 FOR I = 1 TO NOBS
18185 NODS=I-1:IF EOF(1) THEN 18260
18190 INPUT#1,AOBS(I)
18200 IF IUSERWT = 1
 THEN INPUT#1,OBSWT(I)
 ELSE OBSWT(I) = 1#
18210 FOR J = 1 TO NVAR
18220 INPUT#1,VOBS(I,J)
18230 IF IUSERWT = 1
 THEN INPUT#1,VARWT(I,J)
 ELSE VARWT(I,J) = 1#
18240 NEXT J
18250 NEXT I
18260 FOR I = 1 TO NP
18270 INPUT#1,PNAME$(I), P(I)
18280 NEXT I
18285 NOBS=NODS
18290 DEVSQ1 = 1D+20 : DEVSQ2 = 1D+20 'dummy devsqs for
 non-converge test
18300 TIME$ = "00:00:00"
18305 CLOSE #1
18310 RETURN
19010 '* SUBROUTINE REPORT
19030 '* Prints final report giving observed and calculated
 values of
19040 '* function, and standard deviation
19050 '* Note: assumes NEC 8023 printer. Modify to suit other
 printers.
19060 '* On NEC, esc X turns on underlining, esc Y turns it off.
19070 '* The following is a partial list of IBM screen
 characters, followed
19080 '* by the charactor the NEC will print: û = | ; = sum;

19090 '* Thus " error| =" prints as "sum error^ 2 =".
19110 GOSUB 11000 'subroutine function
19130 LPRINT:LPRINT "FINAL REPORT"
19140 LPRINT "FUNCTION: "; FUNCTION$
19150 LPRINT "DATA FILE:";IFILE$
19160 LPRINT : LPRINT" Function Values"
19170 LPRINT "X" "Observed | Calculated"; : LPRINT "Y"
19172 PDRAW$=IFILE$+"r":PDRAW2$=IFILE$+"c"
    PDRAW3$=IFILE$+ "o"
19175 OPEN PDRAW$ FOR OUTPUT AS #3
19176 OPEN PDRAW3$ FOR OUTPUT AS #2
```

```
19177 OPEN PDRAW2$ FOR OUTPUT AS #4
19180 DEVSQ = 0#
19185 L=1
19190 FOR I = 1 TO NOBS
19192 IF VOBS(I,1)<0 THEN VOBS(I,2)=-VOBS(I,2)
19193 L=L+1
19194 IF L>2 THEN L=1
19195 WRITE#3,VOBS(I,1),FCALC2(I)-AOBS(I)
19196 WRITE#4,VOBS(I,2),FCALC2(I)
19197 WRITE#2,VOBS(I,1),AOBS(I)
19200 LPRINT USING"####.##  ####.##";VOBS(I,1);
" | ";VOBS(I,2)
19210 DEVSQ = DEVSQ + FCALC(I)^ 2 * OBSWT(I)*WEIGHT
19220 NEXT I
19225 CLOSE #3: CLOSE #4:CLOSE #2
19230 LPRINT
19240 LPRINT USING "  ####.###########
   +#.###########";" error| = "; DEVSQ;
" Change, last iteration = "; DEVSQ - DEVSQ1
19250 GOSUB 19500 'subroutine covariance
19260 LPRINT: LPRINT"FINAL PARAMETERS"
19270 LPRINT "X" " Name | Value | Std.Dev."; : LPRINT "Y"
19280 FOR I = 1 TO NP
19290 LPRINT USING "   ####.###########
 ##.############"; PNAME$(I);"| "; P(I);" | ";
ABS(B(I,I))^ .5
19292 SURT=-979.699*DENDIF*XSCAL^ 2/P(2):STFFD$="surface tension"
19293 IF I=2 THEN LPRINT USING "   ###.####";
STFFD$;"|";SURT
19300 NEXT I
19310 LPRINT
19320 KEY ON: RETURN
19510 '* SUBROUTINE COVARIANCE
19530 '* Calculates estimate of unit variance, then calculates
19540 '* variance/covariance matrix based on inverted B matrix
and the
19550 '* variance estimate.
19570 'estimate unit variance
19580 IF NOBS <= NP
THEN VAR = 0# 'should never trust parameters if nobs < np
anyways! ELSE VAR = DEVSQ / (NOBS - NP)
19590 'convert B to covariance matrix
19592 VAR=DEVSQ/(NOBS-NP)
19600 FOR I = 1 TO NP
19610 FOR J = 1 TO NP
19620 B(I,J) = B(I,J) * VAR
19630 NEXT J
```

```
19640 NEXT I
19650 RETURN
20010 '* subroutine finalproc
20020 '*
20030 '* Before nonlin stops, it makes a call to a subroutine at
 line 20000
20040 '* The user may supply a subroutine (in the file with the
 function
20050 '* subroutine) to do any final calculations using either
 the final
20060 '* parameter set and/or the variance-covariance matrix.
20080 RETURN
65000 ' Trap error of function file not in ascii mode
65010 IF ERR <> 54 THEN 65090
65020 CLS: BEEP : LOCATE 5,28
65030 PRINT "Bad File Mode Error:"
65040 LOCATE 7,21: PRINT "Function file must be saved in ASCII
 mode"
65050 LOCATE 8,15
65080 LOCATE 23,1: STOP
65090 RESUME
```

NONLI8.BAS: A MICROSOFT BASIC PROGRAM
(Written for the IBM Personal Computer)

This program uses nonlinear parameter estimation algorithms that were developed by Dave Whitman to curvefit experimental pendant drop profile data to the Laplace equation. The algorithm was modified by Michael T. Harris so that analytical derivatives are used to compute the Z-matrix.

The following data which is not contained in the data file must be entered by the user:

1. Number of data files to be processed.
2. Full name of data file.
3. The x-scaling factor, i.e. cm/(measured unit).
4. Density difference between the saturated phases, g/cm^3.

A listing of the program is given below. Lines 11000 – 11900 contains the fourth-order Runge–Kutta method for solving a set of first order differential equations to determine x_{calc} and y_{calc}. Lines 31085 – 41800 contains the fourth-order Runge–Kutta method which is used to compute the analytical derivatives that are required to form the Z–matrix.

```
15 DEFINT I-N
16 DEFDBL A-H,O-Z
110 DIM AOBS(300),FCALC(300),FTEMP(300),OBSWT(300)
120 DIM VOBS(300,2),V(300,2),VARWT(300,2),DFDV(300,2)
130 DIM P(4),PNAME$(4), DFDP(300,4),INDEX(4,5),B(4,4)
140 DIM IFLAG(300),DLAMBDA(300),RHS(4),BCD(4,4),FCALC2(300)
150 DIM IFIL1$(30),XSCA1(30),DENDI1(30),IP(300)
10050 NVAR=1 : NP=2
10052 NFT=3
10055 IYT=1:NONDIV=0:NONCONVERGE=0
10060 GOSUB 18000 'initialization routine'
10065 IFT=IFT+1
10080 FOR IT = 1 TO NUMIT
10090 'print progress report on screen'
10100 GOSUB 12000 'subroutine iteration report'
10110 'Test for non-convergance, exit if so'
10120 GOSUB 13000 'subroutine converge'
10130 IF NONCONVERGE = 1
THEN PRINT"nonconvergence termination" : GOTO 10270
10135 IF NONDIV=1 THEN PRINT "convergence": GOTO 10270
10140 'Calculate lagrangian multipliers'
10150 GOSUB 13500 'subroutine lambda'
10160 'If internal weighting desired, calculate new obswts'
10170 IF INTERNALWT = 1
THEN GOSUB 14500
10180 'Set up matrix equation to get parameter changes'
10190 GOSUB 15000 'SUBROUTINE SETUP'
```

```
10200 'Solve equation for parameter changes'
10210 GOSUB 16000 'subroutine solve'
10220 'Apply changes'
10230 GOSUB 17000 'subroutine deltap'
10240 NEXT IT
10260 'print final report'
10270 GOSUB 19000 'subroutine report'
10280 'Do any final processing (user supplied)'
10290 GOSUB 20000
10295 IF IFT¡=NFT THEN GOTO 10055
10300 END
11000 DS=.013158*P(1)
11010 S=DS:JCOUNT=0:IERR=1
11020 THI1=0
11025 THI1B=0
11027 THI1R=0
11030 THI=THI1
11035 THIB=THI1B
11037 THIR=THI1R
11040 XCALC=0
11045 XCALCB=0
11047 XCALCR=0
11050 YCALC=0
11055 YCALCB=0
11057 YCALCR=0
11070 REM runge-kutta method
11080 J=J
11085 XOLD=XCALC
11090 YOLD=YCALC
11110 THOLD=THI
11120 SP=S/P(1)
11125 SF2=SP:THF2=THOLD
11130 GOSUB 11740
11135 XK1=VF2
11140 XK1=XK1*DS
11145 SF1=SP:THF1=THOLD
11150 GOSUB 11710
11155 XL1=VF1
11160 XL1=XL1*DS
11165 SF1=SP:XF3=XCALC:YF3=YCALC:THF1=THOLD
11170 GOSUB 11770
11175 XM1=VF3
11180 XM1=XM1*DS
11190 SP=(S+DS/2!)/P(1)
11200 XP=XCALC+XK1/2!
11220 YP=YCALC+XL1/2!
11230 TIP=THOLD+XM1/2!
```

```
11235 SF2=SP:THF2=TIP
11240 GOSUB 11740
11245 XK2=VF2
11250 XK2=XK2*DS
11253 SF1=SP:THF1=TIP
11255 GOSUB 11710
11256 XL2=VF1
11257 XL2=XL2*DS
11258 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11260 GOSUB 11770
11265 XM2=VF3
11270 XM2=XM2*DS
11280 XP=XCALC +XK2/2!
11290 YP= YCALC+XL2/2
11310 TIP=THI+XM2/2!
11315 SF2=SP:THF2=TIP
11320 GOSUB 11740
11325 XK3=VF2
11330 XK3=XK3*DS
11335 SF1=SP:THF1=TIP
11340 GOSUB 11710
11345 XL3=VF1
11350 XL3=XL3*DS
11355 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11360 GOSUB 11770
11365 XM3=VF3
11370 XM3=XM3*DS
11380 SP=(S+DS)/P(1)
11390 XP=XCALC +XK3
11410 YP=YCALC + XL3
11420 TIP=THI +XM3
11425 SF2=SP:THF2=TIP
11430 GOSUB 11740
11435 XK4=VF2
11437 SF1=SP:THF1=TIP
11440 GOSUB 11710
11445 XL4=VF1
11447 SF1=SP:XF3=XP:YF3=YP:THF1=TIP
11450 GOSUB 11770
11452 XM4=VF3
11455 GOSUB 31085: GOSUB 41085
11460 S=S+DS
11470 XCALC=XCALC+(XK1+2!*XK2+2!*XK3+XK4*DS)/6!
11475 YCALC=YCALC+(XL1+2!*XL2+2!*XL3+XL4*DS)/6!
11477 REM YCALC=YCALC+(-(XCALC-XOLD)^ 2+DS^ 2)^ .5
11490 THI=THI +(XM1+2!*XM2+2!*XM3+XM4*DS)/6!
11492 IF THI¿3.14 OR THI¡0 GOTO 11580
```

```
11493 IF IMIN¡=1 GOTO 11500
11500 IMIN=IERR
11502 FOR I=1 TO NOBS: IP(I)=I: NEXT I
11505 FOR J=IMIN TO NOBS: I=IP(J)
11510 X0=XOLD:Y0=YOLD:X1=XCALC:Y1=YCALC:XS=ABS(VOBS(I,1)):
YS=ABS(AOBS(I))
11512 REM IF DII=2 THEN XS=VOBS(I,1)+P(3):DII=1
11514 DII=DII+1
11515 DELT=(X1-X0)^ 2+(Y1-Y0)^ 2
11520 IF DELT=0 GOTO 11550
11522 STEST0=((X1-X0)*(XS-X0)+(Y1-Y0)*(YS-Y0))/DELT
11524 STEST1=((X1-X0)*(XS-X1)+(Y1-Y0)*(YS-Y1))/DELT
11526 IF STEST1¡¿0 GOTO 11532
11528 VOBS(I,2)=X1:FCALC(I)=Y1:FCALC2(I)=Y1
11529 FCALC(I)=(XS-VOBS(I,2))^ 2+(YS-FCALC(I))^ 2:
DFDP(I,1)=2*((XCALC-XS)*XCALCR+(YCALC-YS)*YCALCR)
11530 JCOUNT=JCOUNT+1:IERR=I+1:DFDP(I,2)=4*((XCALC-XS)*
XCALCB+(YCALC-YS)*YCALCB):DFDP(I,3)=2*(XCALC-XS):
DFDP(I,4)=2*(YCALC-YS):IP(IMIN)=J:IP(J)=IMIN
11531 LOCATE 1,1:PRINT USING "### ";I,IERR:GOTO 11550
11532 IF STEST0=0 GOTO 11550
11534 STS=SGN(1!-STEST0):STS1=SGN(1!-STEST1)
11536 IF STS=STS1 GOTO 11550
11538 VOBS(I,2)=X0+STEST0*(X1-X0):FCALC(I)=Y0+
   STEST0*(Y1-Y0):FCALC2(I)=FCALC(I)
11539 XCALB1=XOLDB+(VOBS(I,2)-X0)/(X1-X0)*
   (XCALCB-XOLDB):YCALB1=YOLDB+(FCALC2(I)-Y0)/
   (Y1-Y0)*(YCALCB-YOLDB)
11540 JCOUNT=JCOUNT+1
11541 FCALC(I)=(XS-VOBS(I,2))^ 2+(YS-FCALC(I))^ 2:IERR=I+1
11542 XCALR1=XOLDR+(VOBS(I,2)-X0)/(X1-X0)*(XCALCR-XOLDR):
   YCALR1=YOLDR+(FCALC2(I)-Y0)/(Y1-Y0)* (YCALCR-YOLDR)
11543 DFDP(I,2)=4*((VOBS(I,2)-XS)*XCALB1+(FCALC2(I)-YS)*
YCALB1):IP(IMIN)=J
11545 DFDP(I,1)=2*((VOBS(I,2)-XS)*XCALR1+(FCALC2(I)-YS)*
YCALR1):IP(J)=IMIN
11546 DFDP(I,3)=2*(VOBS(I,2)-XS):DFDP(I,4)=2*(FCALC2(I)-YS)
11550 LOCATE 1,1:PRINT USING "### ";I,IERR:NEXT J
11560 IF IERR¡=NOBS GOTO 11080
11580 WEIGHT=(NOBS-2)/(JCOUNT-2)
11700 GOTO 11900
11710 REM sunb f1
11720 VF1=SIN(THF1)
11730 RETURN
11740 REM sun f2.f
11750 VF2=COS(THF2)
11760 RETURN
```

```
11770 REM f3.f
11780 GOSUB 11710
11785 ZF3=VF1
11787 IF XF3=0 THEN VF3=1/P(1):GOTO 11800
11790 VF3=2/P(1)+P(2)*YF3-ZF3/XF3
11800 RETURN
11900 RETURN
12010 'SUBROUTINE ITERTATION REPORT'
12030 'Prints out current parameters, function values, and deviation'
12050 GOSUB 12200 'print parameters'
12060 GOSUB 11000 'get function values in fcalc'
12070 GOSUB 12500 'print function values and deviation'
12080 RETURN
12210 'SUBROUTINE PARAMREPORT( IT, NP, P)'
12220 '2/7/82 by Dave Whitman'
12230 'Prints out current parameter values'
12250 CLS : LOCATE 4,4
12260 PRINT "Parameters, Iteration";IT : PRINT
12265 LPRINT "Parameters, Iteration";IT : LPRINT
12270 LOCATE 7,2
12280 COLOR 1 : PRINT " Name | Value | Change "; : COLOR 7 : PRINT
12285 LPRINT "Name | Value | Change ";:LPRINT
12290 FOR I = 1 TO NP
12300 LOCATE I+7,2
12310 PRINT USING "   #.#####
   ####"; PNAME$(I) ;"|
"; P(I); "| ";-1 * RHS(I)
12311 SURT=-979.699*DENDIF*XSCAL^ 2/P(2):STFFD$="surface tension"
12312 IF I=4 THEN PRINT USING "   ###.####";ST- FFD$;"| ";SURT
12315 LPRINT USING "   ###.####### ####.#########"; PNAME$(I);"|
"; P(I); " | ";-1 * RHS(I)
12316 IF I=2 THEN LPRINT USING "   ###.####";
STFFD$;"—";SURT
12320 NEXT I
12330 PRINT
12335 LPRINT
12340 RETURN
12500 '*******************************************************
12510 '* SUBROUTINE FUNCTIONREPORT
 *
12520 '* Prints obs. and calc. function values, and deviation
between them *
12530 '*******************************************************
12540 IROW = 1 : ICOL = 40 : IOFFSET = 20 : IROOM = 40 : NUMROWS
= 20
12550 LOCATE IROW,ICOL : COLOR 1
12560 PRINT " obs. | calc. ";:COLOR 7 : PRINT
```

```
12565 LPRINT " obs. | calc. ";:LPRINT
12570 IF NOBS ¿= NUMROWS
 THEN LOCATE IROW,ICOL+IOFFSET : COLOR 1:
 PRINT " obs. 3 calc.
"; : COLOR 7 : PRINT
12580 DEVSQ = 0
12590 FOR J = 1 TO NOBS: I=IP(J)
12600 IF I ¿ IROOM THEN 12630
12610 IF I ¡= NUMROWS THEN LOCATE (IROW + I),ICOL
 ELSE LOCATE (IROW + I
MOD NUMROWS),(ICOL + IOFFSET)
12620 PRINT USING "#.####  #.####"; VOBS(I,1)-P(3);" 3
";VOBS(I,2);
12630 DEV = FCALC(I)*WEIGHT
12640 DEVSQ = DEVSQ + DEV * OBSWT(I)
12650 NEXT J
12660 LOCATE 20,5 : PRINT USING "  ##.#####";"sum error^ 2 =
";DEVSQ
12665 LPRINT USING "  ##.#####";"sum error^ 2 = ";DEVSQ
12670 IF IT = 1 THEN 12690 'no change in first iteration
12680 LOCATE 21,5: PRINT USING "  #.######";"Change = ";DEVSQ-DEVSQ1;
12685 LPRINT USING "  #.######";" Change = ";DEVSQ-DEV-SQ1;
12690 RETURN
13000 '*******************************************************
13010 '* SUBROUTINE CONVERGE ( ERRSQ, DEVSQ, DEVSQ1, DEVSQ2,
NONCONVERGE ) *
13020 '*
 *
13030 '* Compares squared deviation of calculated function from
observed *
13040 '* function with that obtained in the last 2 iterations. If
the *
13050 '* deviation got worse two iterations in a row, set
nonconverge flag.*
13060 '*******************************************************
13070 IF (DEVSQ ¿ 1.05*DEVSQ1 AND DEVSQ1 ¿ 1.05*DEVSQ2)
 THEN NONCONVERGE = 1
13075 IF ABS(DEVSQ-DEVSQ1)/DEVSQ¡.0001 THEN NONDIV=1
13080 DEVSQ2 = DEVSQ1
13090 DEVSQ1 = DEVSQ
13100 RETURN
13500 '*******************************************************
13510 '* SUBROUTINE LAMBDA ( DLAMBDA, FCALC, FOBS, OBSWT)
 *
13520 '*
 *
13530 '* Calculates lagrangian multipliers for setting up matrix
```

```
equation *
13540 '*******************************************************
13550 FOR J = 1 TO NOBS: I=IP(J)
13560 DLAMBDA(I) = FCALC(I) * OBSWT(I)*WEIGHT
13570 NEXT J
13580 RETURN
14000 '*******************************************************
14010 '* SUBROUTINE VSLOPE( V,DFDV,NVAR )
 *
14020 '*
 *
14030 '* Calculates the partials of the function w/ r.t. each
 *
14040 '* of the variables at each of the observed points, and
 *
14050 '* stores them in dfdv.
 *
14060 '*******************************************************
14070 GOSUB 11000 'call function( p, v, nobs, nvar, np, fcalc)
14080 FOR IW = 1 TO NOBS
14090 FTEMP(IW) = FCALC(IW)
14100 NEXT IW
14110 FOR IW = 1 TO NVAR
14120 FOR JW = 1 TO NOBS
14130 IFLAG(JW) = 0
14140 IF ABS(VOBS(JW,IW)) ¡ 1D-20
 THEN VOBS(JW,IW) = .0005# :
IFLAG(JW) = 1 ELSE VOBS(JW-
,IW) = VOBS(JW,IW) * 1.0005#
14150 PRINT "modified var:" ;VOBS(JW,IW)
14160 NEXT JW
14170 GOSUB 11000 'call function(fcalc)
14180 FOR JW = 1 TO NOBS
14190 IF IFLAG(JW) = 1
 THEN DFDV(JW,IW) = (FCALC(JW) -
FTEMP(JW)) / .0005# : VOBS(JW,IW)
= VOBS(JW,IW) - .0005#
14200 IF IFLAG(JW) ¡¿ 1
 THEN DFDV(JW,IW) = (FCALC(JW)-FTEM-
P(JW)) / (.0005# * VOBS(JW,IW)) : VOBS(JW,IW)
= VOBS(JW,IW) / 1.0005#
14210 PRINT"dfdv(";JW;IW;")="; DFDV(JW,IW)
14220 NEXT JW
14230 NEXT IW
14240 RETURN
14500 '*******************************************************
14510 '* subroutine weigh[ p, nobs, nvar, v, dfdv, varwt, obswt ]
```

```
*
14520 '*
*
14530 '* calculates new weights for function values,
*
14540 '* using the follwing formula:
*
14550 '* obswt(i) = 1/sum [(dfdv)^ 2 * (1/varwt(v))]
*
14560 '* note: obswt(i) = 1/L(i) in Wentworth article
*
14570 '*******************************************************
14580 IF IT = 1 THEN RETURN 'skip weighting on first iteration
14590 GOSUB 14000 'subroutine vslope
14600 FOR IW = 1 TO NOBS
14610 SUM = 0#
14620 FOR JW = 1 TO NVAR
14630 SUM = SUM + DFDV(IW,JW)*DFDV(IW,JW)/VARWT(IW,JW)
14640 NEXT JW
14650 OBSWT(IW) = 1# / SUM
14660 NEXT IW
14670 PRINT "new function weights:"
14680 FOR IW = 1 TO NOBS
14690 PRINT OBSWT(IW)
14700 NEXT IW
14710 RETURN
15000 '*******************************************************
15010 '* SUBROUTINE SETUP( B,RHS,dfdp) *
15020 '* Sets up matrix equation to get changes to parameters *
15030 '*******************************************************
15040 '
15050 'Get partials of function w.r.t. parameters
15060 GOSUB 17500 'subroutine pslope
15070 'Now set up matrices
15080 FOR I = 1 TO NP
15090 'Set up right hand side element
15100 RHS(I) = 0#
15110 FOR JJ = 1 TO NOBS: J=IP(JJ)
15120 RHS(I) = RHS(I) + DFDP(J,I) * DLAMBDA(J)
15130 NEXT JJ
15140 'Set up left hand side elements
15150 FOR J = 1 TO NP
15160 B(I,J) = 0#
15170 FOR KK = 1 TO NOBS:K=IP(KK)
15180 B(I,J) = B(I,J) + DFDP(K,I) * DFDP(K,J) *
OBSWT(K)
15190 NEXT KK
```

```
15200 NEXT J
15210 NEXT I
15220 RETURN
16000 '*****************************************************
16010 '* subroutine solve[b#(np,np), rhs(np), np] *
16020 '* 1/31/82 by Dave Whitman *
16030 '* solves matrix equations of the form b# x = rhs# *
16040 '* inverts b# in place,multiplies rhs# by inverse *
16050 '* uses Gauss-Jordan matrix inversion *
16060 '* for good results b# and rhs# must be dbl precision*
16070 '* ref: J.M. McCormick "Numerical Methods in FORTRAN"*
16080 '*****************************************************
16090 DETERM = 1#
16100 FOR I = 1 TO NP
16110 INDEX(I,3) = 0
16120 NEXT I
16130 FOR I = 1 TO NP 'MAIN LOOP
16140 'search for pivot element
16150 MAX# = 0#
16160 FOR J = 1 TO NP
16170 IF INDEX(J,3) = 1 THEN 16260
16180 FOR K = 1 TO NP
16190 IF INDEX(K,3) ¿ 1 THEN 16700
16200 IF INDEX(K,3) = 1 THEN 16250
16210 IF MAX# ¿ ABS(B(J,K)) THEN 16250
16220 IROW = J
16230 ICOL = K
16240 MAX# = ABS(B(J,K))
16250 NEXT K
16260 NEXT J
16270 INDEX(ICOL,3) = INDEX(ICOL,3) + 1
16280 INDEX(I,1) = IROW
16290 INDEX(I,2) = ICOL
16300 'interchange rows to put pivot on diagonal
16310 IF IROW = ICOL THEN 16380 'ALREADY THERE
16320 DETERM = -1# * DETERM
16330 FOR J = 1 TO NP
16335 PRINT "brow= ";B(IROW,J);" bcol= ";B(ICOL,J)
16340 SWAP B(IROW,J),B(ICOL,J)
16345 PRINT "brow= ";B(IROW,J);" bcol= ";B(ICOL,J)
16350 NEXT J
16355 PRINT "rrow= ";RHS(IROW);" rcol= ";RHS(ICOL)
16360 SWAP RHS(IROW),RHS(ICOL)
16370 'divide pivot row by pivot element
16380 PIVOT = B(ICOL,ICOL)
16390 DETERM = DETERM * PIVOT
16400 B(ICOL,ICOL) = 1#
```

```
16410 FOR J = 1 TO NP
16420 B(ICOL,J) = B(ICOL,J)/PIVOT
16430 NEXT J
16440 RHS(ICOL) = RHS(ICOL)/PIVOT
16450 ' reduce non-pivot rows
16460 FOR J = 1 TO NP
16470 IF J = ICOL THEN 16540
16480 T = B(J,ICOL)
16490 B(J,ICOL) = 0# :BCD(J,ICOL)=B(J,ICOL)
16500 FOR K = 1 TO NP
16510 B(J,K) = B(J,K) - B(ICOL,K)*T:BCD(J,K)=B(J,K)
16520 NEXT K
16530 RHS(J) = RHS(J) - RHS(ICOL)*T
16540 NEXT J
16550 NEXT I
16560 'interchange columns
16570 FOR I = NP TO 1 STEP -1
16580 IF INDEX(I,1) = INDEX(I,2) THEN 16640
16590 IROW = INDEX(I,1)
16600 ICOL = INDEX(I,2)
16610 FOR J = 1 TO NP
16620 SWAP B(J,IROW), B(J,ICOL)
16630 NEXT J
16640 NEXT I
16650 'test for singularity
16660 FOR I = 1 TO NP
16670 IF INDEX(I,3) ¡¿ 1 THEN 16700
16680 NEXT I
16690 RETURN
16700 PRINT"singular matrix error ":RETURN
17000 '*********************************************************
17010 '* SUBROUTINE DELTAP ( P, RHS, NP ) *
17020 '* *
17030 '* Modifies parameters according to changes in rhs *
17040 '*********************************************************
17050 FOR I = 1 TO NP
17060 P(I) = P(I) - RHS(I) * FRACT
17070 NEXT I
17080 RETURN
17500 '*********************************************************
17510 '* subroutine pslope[ p, vobs, nobs, np, nvar, dfdp(nobs,np) ] *
17520 '* 2/1/82 by Dave Whitman
 *
17530 '* calculates partial of the function with
 *
17540 '* respect to the each of the parameters at
 *
```

```
17550 '* each of the observations, and stores them in dfdp.
 *
17560 '*******************************************************
17580 FOR ISS = 1 TO NOBS:IS=IP(ISS)
17590 FTEMP(IS) = FCALC(IS)*WEIGHT
17600 NEXT ISS
17610 FOR IS = 1 TO NP
17620 TP = P(IS)
17625 P(IS) = TP*1.000000005#
17630 REM iF TP ¡ 1D-20
 THEN P(IS) = TP + .000000005#
 ELSE P(IS) =
TP * 1.000000005#
17640 GOTO 17650: GOSUB 11000 'call function( fcalc )
17645 DPP=0
17650 FOR JSS = 1 TO NOBS:JS=IP(JSS)
17660 REM IF TP ¡ 1D-20 THEN DFDP(JS,IS) = (FCALC(JS)*WEIGHT -
FTEMP(JS)) / .000000005# ELSE DFDP(JS,IS) = (FCALC(JS)*WEIGHT -
FTEMP(JS)) / (.000000005# * TP)
17665 DFDP(JS,IS)=DFDP(JS,IS)*WEIGHT
17670 NEXT JSS
17675 REM FOR JS=1 TO NOBS
17676 REM DFDP(JS,IS)=(1-DPP/DEVSQ)/(ABS(1-DPP/DEVSQ))
    *DFDP(JS,IS)
17677 REM NEXT JS
17680 P(IS) = TP
17690 NEXT IS
17700 RETURN
18000 '*******************************************************
18010 '* SUBROUTINE initialize
 *
18020 '*
 *
18030 '* Prompts for name of input file, then reads problem in
 *
18040 '* from input file.
 *
18050 '*******************************************************
18060 '
18070 KEY OFF : CLS
18075 IF IFT ¿1 THEN 18086
18076 IFT=1: INPUT "enter # of files";NFT
18077 FOR I=1 TO NFT
18080 LOCATE 13,20 : INPUT "Name of input file? ",IFIL1$(I)
18082 INPUT "enter x-scale and density diff";XSCA1(I),DENDI1(I)
18084 NEXT I
18086 IFILE$=IFIL1$(IFT):XSCAL=XSCA1(IFT):DENDIF=DENDI1(IFT)
```

```
18090 OPEN IFILE$ FOR INPUT AS #1
18100 INPUT#1,NOBS
18150 INPUT#1,NUMIT
18160 INPUT#1,IUSERWT,INTERNALWT
18170 INPUT#1, FRACT 'fraction of calculated param. change to
apply
18180 FOR I = 1 TO NOBS
18185 NODS=I-1:IF EOF(1) THEN 18260
18190 INPUT#1,AOBS(I)
18200 IF IUSERWT = 1
 THEN INPUT#1,OBSWT(I)
 ELSE OBSWT(I) = 1#
18210 FOR J = 1 TO NVAR
18220 INPUT#1,VOBS(I,J)
18230 IF IUSERWT = 1
 THEN INPUT#1,VARWT(I,J)
 ELSE VARWT(I,J) = 1#
18240 NEXT J
18250 NEXT I
18260 FOR I = 1 TO NP
18270 INPUT#1,PNAME$(I), P(I)
18280 NEXT I
18285 NOBS=NODS
18290 DEVSQ1 = 1D+20 : DEVSQ2 = 1D+20 'dummy devsqs for non-conv-
erge test
18300 TIME$ = "00:00:00"
18305 CLOSE #1
18310 RETURN
19000 '*******************************************************
19010 '* SUBROUTINE REPORT
 *
19020 '*
 *
19030 '* Prints final report giving observed and calculated
values of *
19040 '* function, and standard deviation
 *
19050 '* Note: assumes NEC 8023 printer. Modify to suit other
printers. *
19060 '* On NEC, esc X turns on underlining, esc Y turns it off.
 *
19070 '* The following is a partial list of IBM screen charactor-
s, followed *
19080 '*
19090 '* Thus "sum error ^ 2 =" prints as "sum error ^ 2 =".
 *
19100 '*******************************************************
```

```
19110 GOSUB 11000 'subroutine function
19130 LPRINT:LPRINT "FINAL REPORT"
19140 LPRINT "FUNCTION: "; FUNCTION$
19150 LPRINT "DATA FILE:";IFILE$
19160 LPRINT : LPRINT" Function Values"
19170 LPRINT "X" "Observed | Calculated"; : LPRINT "Y"
19172 PDRAW$=IFILE$+"r":PDRAW2$=IFILE$+"c":PDRAW3$= IFILE$+"o"
19175 OPEN PDRAW$ FOR OUTPUT AS #3
19176 OPEN PDRAW3$ FOR OUTPUT AS #2
19177 OPEN PDRAW2$ FOR OUTPUT AS #4
19180 DEVSQ = 0#
19185 L=1
19190 FOR II = 1 TO NOBS:I=IP(II)
19195 WRITE#3,VOBS(I,1)-P(3),FCALC2(I)-AOBS(I)+P(4)
19196 WRITE#4,VOBS(I,2),FCALC2(I)
19197 WRITE#2,VOBS(I,1)-P(3),AOBS(I)-P(4)
19200 LPRINT USING"####.##  ####.##";VOBS(I,1)-P(3);" |
";VOBS(I,2)
19210 DEVSQ = DEVSQ + FCALC(I) * OBSWT(I)*WEIGHT
19220 NEXT II
19225 CLOSE #3: CLOSE #4:CLOSE #2
19230 LPRINT
19240 LPRINT USING "  ####.###########
 #.###########"; "sum error ^  2 = "; DEVSQ;
" Change, last iteration = "; DEVSQ - DEVSQ1
19250 GOSUB 19500 'subroutine covariance
19260 LPRINT: LPRINT"FINAL PARAMETERS"
19270 LPRINT "X" " Name | Value | Std.Dev."; : LPRINT "Y"
19280 FOR I = 1 TO NP
19290 LPRINT USING "   ####.###########  ##.########
####"; PNAME$(I);"| "; P(I);" | "; ABS(B(I,I))^ .5
19292 SURT=-979.699*DENDIF*XSCAL^ 2/P(2):STFFD$="surface tension"
19293 IF I=2 THEN LPRINT USING "   ###.####";
   STFFD$;"|";SURT
19300 NEXT I
19310 LPRINT
19320 KEY ON: RETURN
19500 '*******************************************************
19510 '* SUBROUTINE COVARIANCE
 *
19520 '*
 *
19530 '* Calculates estimate of unit variance, then calculates
 *
19540 '* variance/covariance matrix based on inverted B matrix
and the *
19550 '* variance estimate.
```

```
 *
19560 '*******************************************************
19570 'estimate unit variance
19580 IF NOBS ¡= NP
 THEN VAR = 0# 'should never trust
parameters if nobs ¡ np anyways! ELSE VAR = DEVSQ /
(NOBS - NP)
19590 'convert B to covariance matrix
19592 VAR=DEVSQ/(NOBS-NP)
19600 FOR I = 1 TO NP
19610 FOR J = 1 TO NP
19620 B(I,J) = B(I,J) * VAR
19630 NEXT J
19640 NEXT I
19650 RETURN
20000 '*******************************************************
20010 '* subroutine finalproc
 *
20020 '*
 *
20030 '* Before nonlin stops, it makes a call to a subroutine at
line 20000 *
20040 '* The user may supply a subroutine (in the file with the
function *
20050 '* subroutine) to do any final calculations using either
    the final *
20060 '* parameter set and/or the variance-covariance matrix.
     *
20070 '*******************************************************
20080 RETURN
31085 XOLDB=XCALCB
31090 YOLDB=YCALCB
31110 THOLDB=THIB
31125 SF2=SP:THF2B=THOLDB:THF2=THOLD
31130 GOSUB 31740
31135 XK1B=VF2B
31140 XK1B=XK1B*DS
31145 SF1=SP:THF1B=THOLDB:THF1=THOLD
31150 GOSUB 31710
31155 XL1B=VF1B
31160 XL1B=XL1B*DS
31165 SF1=SP:XF3B=XCALCB:YF3B=YCALCB:THF1B=THOLDB:XF3=XCALC:
    YF3=YCALC:THF1=THOLD
31170 GOSUB 31770
31175 XM1B=VF3B
31180 XM1B=XM1B*DS
31190 SP=(S+DS/2!)/P(1)
```

```
31200 XPB=XCALCB+XK1B/2!
31220 YPB=YCALCB+XL1B/2!
31230 TIPB=THOLDB+XM1B/2!
31235 SF2=SP:THF2B=TIPB:THF2=THOLD+XM1/2
31240 GOSUB 31740
31245 XK2B=VF2B
31250 XK2B=XK2B*DS
31253 SF1=SP:THF1B=TIPB:THF1=THOLD+XM1/2
31255 GOSUB 31710
31256 XL2B=VF1B
31257 XL2B=XL2B*DS
31258 SF1=SP:XF3B=XPB:YF3B=YPB:THF1B=TIPB:XF3=XCALC+XK1/2:
   YF3=XCALC+XL1/2:THF1=THOLD+XM1/2
31260 GOSUB 31770
31265 XM2B=VF3B
31270 XM2B=XM2B*DS
31280 XPB=XCALCB+XK2B/2!
31290 YPB= YCALCB+XL2B/2
31310 TIPB=THIB+XM2B/2!
31315 SF2=SP:THF2B=TIPB:THF2=THOLD+XM2/2
31320 GOSUB 31740
31325 XK3B=VF2B
31330 XK3B=XK3B*DS
31335 SF1=SP:THF1B=TIPB:THF1=THOLD+XM2/2
31340 GOSUB 31710
31345 XL3B=VF1B
31350 XL3B=XL3B*DS
31355 SF1=SP:XF3B=XPB:YF3B=YPB:THF1B=TIPB:XF3=XCALC+XK2/2:
   YF3=YCALC+XL2/2:THF1=THOLD+XM2/2
31360 GOSUB 31770
31365 XM3B=VF3B
31370 XM3B=XM3B*DS
31380 SP=(S+DS)/P(1)
31390 XPB=XCALCB+XK3B
31410 YPB=YCALCB + XL3B
31420 TIPB=THIB+XM3B
31425 SF2=SP:THF2B=TIPB:THF2=THOLD+XM3
31430 GOSUB 31740
31435 XK4B=VF2B
31437 SF1=SP:THF1B=TIPB:THF1=THOLD+XM3
31440 GOSUB 31710
31445 XL4B=VF1B
31447 SF1=SP:XF3B=XPB:YF3B=YPB:THF1B=TIPB:XF3=XCALC+XK3:
   YF3=YCALC+XL3:THF1=THOLD+XM3
31450 GOSUB 31770
31465 XM4B=VF3B
31470 XCALCB=XCALCB+(XK1B+2!*XK2B+2!*XK3B+XK4B*DS)/6!
```

```
31475 YCALCB=YCALCB+(XL1B+2!*XL2B+2!*XL3B+XL4B*DS)/6!
31490 THIB=THIB +(XM1B+2!*XM2B+2!*XM3B+XM4B*DS)/6!
31500 RETURN
31710 REM sunb f1
31720 VF1B=THF1B*COS(THF1)
31730 RETURN
31740 REM sun f2.f
31750 VF2B=-THF2B*SIN(THF2)
31760 RETURN
31770 REM f3.f
31780 GOSUB 31710
31785 ZF3B=VF1B: IF XF3=0 THEN VF3B=0: GOTO 31800
31790 VF3B=SIN(THF1)/XF3^ 2*XF3B+YF3+P(2)*YF3B-COS(THF1)/XF3*THF1B
31800 RETURN
41085 XOLDR=XCALCR
41090 YOLDR=YCALCR
41110 THOLDR=THIR
41125 SF2=SP:THF2R=THOLDR:THF2=THOLD
41130 GOSUB 41740
41135 XK1R=VF2R
41140 XK1R=XK1R*DS
41145 SF1=SP:THF1R=THOLDR:THF1=THOLD
41150 GOSUB 41710
41155 XL1R=VF1R
41160 XL1R=XL1R*DS
41165 SF1=SP:XF3R=XCALCR:YF3R=YCALCR:THF1R=THOLDR:XF3=XCALC:
YF3=YCALC:THF1=THOLD
41170 GOSUB 41770
41175 XM1R=VF3R
41180 XM1R=XM1R*DS
41190 SP=(S+DS/2!)/P(1)
41200 XPR=XCALCR+XK1R/2!
41220 YPR=YCALCR+XL1R/2!
41230 TIPR=THOLDR+XM1R/2!
41235 SF2=SP:THF2R=TIPR:THF2=THOLD+XM1/2
41240 GOSUB 41740
41245 XK2R=VF2R
41250 XK2R=XK2R*DS
41253 SF1=SP:THF1R=TIPR:THF1=THOLD+XM1/2
41255 GOSUB 41710
41256 XL2R=VF1R
41257 XL2R=XL2R*DS
41258              SF1=SP:XF3R=XPR:YF3R=YPR:THF1R=TIPR:XF3=XCALC+XK1/2:
YF3=XCALC+XL1/2:THF1=THOLD+XM1/2
41260 GOSUB 41770
41265 XM2R=VF3R
41270 XM2R=XM2R*DS
```

```
41280 XPR=XCALCR+XK2R/2!
41290 YPR= YCALCR+XL2R/2
41310 TIPR=THIR+XM2R/2!
41315 SF2=SP:THF2R=TIPR:THF2=THOLD+XM2/2
41320 GOSUB 41740
41325 XK3R=VF2R
41330 XK3R=XK3R*DS
41335 SF1=SP:THF1R=TIPR:THF1=THOLD+XM2/2
41340 GOSUB 41710
41345 XL3R=VF1R
41350 XL3R=XL3R*DS
41355 SF1=SP:XF3R=XPR:YF3R=YPR:THF1R=TIPR:XF3=XCALC+XK2/2:
   YF3=YCALC+XL2/2:THF1=THOLD+XM2/2
41360 GOSUB 41770
41365 XM3R=VF3R
41370 XM3R=XM3R*DS
41380 SP=(S+DS)/P(1)
41390 XPR=XCALCR+XK3R
41410 YPR=YCALCR + XL3R
41420 TIPR=THIR+XM3R
41425 SF2=SP:THF2R=TIPR:THF2=THOLD+XM3
41430 GOSUB 41740
41435 XK4R=VF2R
41437 SF1=SP:THF1R=TIPR:THF1=THOLD+XM3
41440 GOSUB 41710
41445 XL4R=VF1R
41447                SF1=SP:XF3R=XPR:YF3R=YPR:THF1R=TIPR:XF3=XCALC+XK3:
YF3=YCALC+XL3:THF1=THOLD+XM3
41450 GOSUB 41770
41465 XM4R=VF3R
41470 XCALCR=XCALCR+(XK1R+2!*XK2R+2!*XK3R+XK4R*DS)/6!
41475 YCALCR=YCALCR+(XL1R+2!*XL2R+2!*XL3R+XL4R*DS)/6!
41490 THIR=THIR +(XM1R+2!*XM2R+2!*XM3R+XM4R*DS)/6!
41500 RETURN
41710 REM sunb f1
41720 VF1R=THF1R*COS(THF1)
41730 RETURN
41740 REM sun f2.f
41750 VF2R=-THF2R*SIN(THF2)
41760 RETURN
41770 REM f3.f
41780 GOSUB 41710
41785 ZF3R=VF1R:IF XF3=0 THEN VF3R=-1/P(1)^ 2:GOTO 41800
41790 VF3R=-2!/P(1)^ 2 + P(2)*YF3R+SIN(THF1)/XF3^ 2*XF3R-THF1R*COS-
(THF1)/XF3
41800 RETURN
65000 ' Trap error of function file not in ascii mode
```

```
65010 IF ERR ¡¿ 54 THEN 65090
65020 CLS: BEEP : LOCATE 5,28
65030 PRINT "Bad File Mode Error:"
65040 LOCATE 7,21: PRINT "Function file must be saved in ASCII
mode"
65050 LOCATE 8,15
65060 PRINT "Read lines 1200-1260 of this program for clarifi-
cation."
65080 LOCATE 23,1: STOP
65090 RESUME
```

PENDR.F: A FORTRAN 77 COMPUTER PROGRAM

This program is the FORTRAN 77 version of the BASIC computer program NONLI4.BAS. Both are used to curvefit experimental pendant drop profile data to the Laplace equation of capillarity.

The following data must be entered by the user:

1. Name of output file.
2. Number of data files to be processed.
3. Full Name of data file.
4. The x-scaling factor, i.e. cm/(measured unit).
5. Density difference between the saturated phases., g/cm3.

A listing of the program is given below.

```
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
      character*80 ifil1
      character*80 pname
      character*80 ifile
      common /main/ nvar,np,nft, numit,fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
      common /chad/ ifile,ifil1(30), pname(4)
c       This program performs non-linear least squares analysis
      to
c       determine surface tension from pendant drop profile data.
c       based on "Rigorous Least Squares Adjustment"; Wentworth,
      W. E.
c       J . Chem Ed. 42, 96 (1965).
c       When nonlin prompts: INPUT FILE?, enter the name of the
      data
c       * file.
c
c       Required data:
c       nobs: the number of observations being fit
c       numit: the # of iterations of the fitting process
      to be
c       performed. [ 5-10 is generally sufficient ]
c       iuserwt: a flag. If iuserwt = 1, nonlin expects all
      obs
c       function and variable values to be
      followed by
c       weighting factor. If iuserwt = 0, nonlin
      automatically
c       sets all initial weights to 1.
c       internalwt: a flag. If internalwt = 1, nonlin
```

```
      estimates
c       weighting factors for each function value
      base
c       the slope of the function. If internalwt
      = 0,
c
c       causes divergence. Start with internalwt =
     0.
c       fract: The fraction of the calculated change to
      apply to
c       of the parameters. Use to restrict changes
      when function
c       trys to diverge. Normally equal to 1.
      NVAR=1
      np=4
     ift=1
      NFT=3
1005   IYT=1
      nond=0
      nonc=0
c       initialization routine
      call suinit(ift)
      IFT=IFT+1
      do 1 it=1,numit
     call calc
c       Test for non-convergance, exit if so
      devb=devsq
      devb1=devsq1
      devb2=devsq2
      call scon(devb,nonc,nond, devb1,devb2)
     devsq1=devb1
     devsq2=devb2
     if(nonc.eq.1) go to 1027
      if(nond.eq.1) go to 1027
c       Calculate lagrangian multipliers
      call lamba
c       If internal weighting desired, calculate new obswts
c       Set up matrix equation to get parameter changes
      call setup
c       Solve equation for parameter changes
      call solve
c       Apply changes
      call deltap
      continue
c
c       print final report
1027   call report(it,nonc,nond)
```

```
      if(nonc.eq.1.or.nond.eq.1) go to 1057
c       Do any final processing (user supplied)
c
 336    continue
 1     continue
1057   continue
      if(ift.le.nft)go to 1005
      stop
      end
      subroutine calc
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
       common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
       common /zxa/ obswt(300),vobs(300,2), v(300,2)
       common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
       common /zxc/ xscal,dendif, xscal(30),dendi1(30)
       common /zxd/ fcalc(300), ftemp(300),varwt(300,2), bcd(4,4)
       common /zxe/ fcalc2(300)
       common/main/ nvar,np,nft,numit, fract
       common /main2/ devsq1,devsq2,nobs, devsq,weight
       continue
       ds=0.01*p(1)
      s=0
       r0=p(1)
       beta=p(2)
       devsq=0
      imin=1
       jcount=0
       ierr=1
       THI1=0
       THI=THI1
       XCALC=0
       YCALC=0
c       REM runge-kutta method
1108   J=J
       XOLD=XCALC
       YOLD=YCALC
       THOLD=THI
       SP=S/P(1)
       SF2=SP
       thf2=thold
       call f2(sp,thi,xk1)
       XK1=XK1*DS
       call f1(sp,thi,xl1)
       XL1=XL1*DS
       call f3(sp,xcalc,ycalc,thi,xm1 ,r0,beta)
       XM1=XM1*DS
```

```
      SP=(S+DS/2.)/p(1)
      XP=XCALC+XK1/2.
      YP=YCALC+XL1/2.
      Tp =THOLD+XM1/2.
      call f2(sp,tp,xk2)
      XK2=XK2*DS
      call f1(sp,tp,xl2)
      XL2=XL2*DS
      call f3(sp,xp,yp,tp,xm2,r0,beta)
      XM2=XM2*DS
      XP=XCALC +XK2/2.
      YP= YCALC+XL2/2.
      Tp =THI+XM2/2.
      call f2(sp,tp,xk3)
      XK3=XK3*DS
      call f1(sp,tp,xl3)
      XL3=XL3*DS
      call f3(sp,xp,yp,tp,xm3,r0,beta)
      XM3=XM3*DS
      SP=(S+DS)/P(1)
      XP=XCALC +XK3
      YP=YCALC + XL3
      Tp =THI +XM3
      call f2(sp,tp,xk4)
      call f1(sp,tp,xl4)
      call f3(sp,xp,yp,tp,xm4,r0,beta)
      S=S+DS
      XCALC=XCALC+(XK1+2.*XK2+2.*XK3+ XK4*DS)/6.
      YCALC=YCALC+(XL1+2.*XL2+2.*XL3 +XL4*DS)/6.
      THI=THI +(XM1+2.*XM2+2.*XM3+XM4* DS)/6.
      IF (thi.gt.3.14.or.thi.lt.0) go to 1158
      IF(imin.le.1) go to 1150
      IMIN=IERR
1150  do 7 i=imin,nobs
      X0=XOLD
      y0=yold
      x1=xcalc
      y1=ycalc
      xs=vobs(i,1)-p(3)
     if(xs.lt.0.0)xs=abs(vobs(i,1)- p(3))
      ys=aobs(i)-p(4)
      DII=DII+1
      DELT=(X1-X0)**2+(y1-y0)**2
      if(delt.eq.0) go to 1155
      STEST0=((X1-X0)*(XS-X0)+(Y1-Y0)* (YS-Y0))/DELT
      STEST1=((X1-X0)*(XS-X1)+(Y1-Y0)* (YS-Y1))/DELT
      IF(stest1.ne.0) go to 1153
```

```
      VOBS(I,2)=X1
      fcalc(i)=y1
      fcalc2(i)=y1
      FCALC(I)=((XS-VOBS(I,2))**2+ (ys-fcalc(i))**2)**.5
      devsq=fcalc(i)**2+devsq
      JCOUNT=JCOUNT+1
      ierr=i+1
     go to 1155
1153   IF(stest0.eq.0)go to 1155
      STS=Sign(1.,stest0)
      sts1=sign(1.,stest1)
      IF (sts.eq.sts1) go to 1155
      VOBS(I,2)=X0+STEST0*(X1-X0)
      fcalc(i)=y0+stest0*(y1-y0)
      fcalc2(i)=fcalc(i)
      FCALC(I)=((XS-VOBS(I,2))**2+ (ys-fcalc(i))**2)**.5
      devsq=fcalc(i)**2+devsq
      ierr=i+1
      JCOUNT=JCOUNT+1
1155   continue
7     continue
      if(ierr.le.nobs) go to 1108
1158   WEIGHT=1.*(NOBS-2)/(1.*JCOUNT-2)
     devsq=devsq*weight
      return
      end
c       subroutine f1
      subroutine f1 (s,th,vg)
     implicit double precision(a-f,o-z)
     implicit integer(g-n)
      vg =SIN(TH)
      RETURN
      end
c       subroutine f2
      subroutine f2 (s,th,vg)
     implicit double precision(a-f,o-z)
     implicit integer(g-n)
      vg =COS(TH)
      return
      end
c       subroutine f3
      subroutine f3 (s,x,y,th,vg,r0, beta)


     implicit double precision(a-f,o-z)
     implicit integer(g-n)
      call f1(s,th,z)
```

```
      if(x.eq.0)vg=1./r0
      if(x.eq.0)go to 2
      vg =2./r0+beta *Y -Z /X
2     continue
      return
      end
c      SUBROUTINE ITERTATION REPORT
c
c      Prints out current parameters, function values, and
     deviatio
c      n
c      ************************** ********************* *******
c      ************************** ********************** *****
c      SUBROUTINE CONVERGE
c      Compares squared deviation of calculated function from
     observed
c      If deviation gets worse after two iterations, indicate
     nonconverge
c      ************************** ********************** *****
      subroutine scon(devv,nonc,nond, devv1,devv2)
     implicit integer (g-n)
     implicit double precision (a-f,o-z)
      IF (devv.gt.devv1.and. devv1.gt.devv2) nonc = 1
      if(abs(devv-devv1)/devv.lt.0.0001) nond =1
      devv2=devv1
      devv1 = devv
      RETURN
      end
c      ************************** ******************* *******
c      SUBROUTINE LAMBDA
c
c      Calculates lagrangian multipliers for setting up matrix
c      equation
c      ************************** ******************* *******
      subroutine lamba
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
      common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xsca1(30),dendi1(30)
      common /zxd/ fcalc(300), ftemp(300),varwt(300,2), bcd(4,4)
     0 common /zxe/ fcalc2(300)
      common /main/ nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
      do 8 i=1,nobs
      DLAMda(i) = FCALC(I) * OBSWT(I)*WEIGHT
```

```
8     continue
      RETURN
      end
c      Subroutine setup
c      ****************************** ********************** ******
c      Get partials of function w.r.t. parameters
      subroutine setup
      implicit integer(g-n)
      implicit double precision(a-f,o-z)
      common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xscal(30),dendi1(30)
      common /zxd/ fcalc(300), ftemp(300),varwt(300,2), bcd(4,4)
      common /zxe/ fcalc2(300)
      common /main/ nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
      call pslope
c      Set up matrices
      do 9 i=1,np
c      Set up right hand side element
      RHS(I) = 0.
      do 10 j=1,nobs
      RHS(I) = RHS(I) + DFDP(J,I) * DLAMda(j)
10    continue
c      Set up left hand side elements
      do 11 j=1,np
      B(I,J) = 0.
      do 12 k=1,nobs
      B(I,J) = B(I,J) + DFDP(K,I) * DFDP(K,J) *
      OBSWT(K)
12    continue
11    continue
9     continue
      RETURN
      end
c      ****************************** ********************** *
c      subroutine solve[b#(np,np), rhs(np), np]
c      1/31/82 by Dave Whitman
c      solves matrix equations of the form b# x = rhs#
c      * inverts b# in place,multiplies rhs# by inverse
c      uses Gauss-Jordan matrix inversion
c      for good results b# and rhs# must be dbl precision
c      ref: J.M. McCormick"Numerical Methods in FORTRAN"
c      ****************************** ******************** **
      subroutine solve
      implicit integer (g-n)
```

```
      implicit double precision (a-f,o-z)
      common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xscal(30),dendil(30)
      common /zxd/ fcalc(300), ftemp(300),varwt(300,2), bcd(4,4)
      common /zxe/ fcalc2(300)
      common /main/ nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
      determ=1.
      do 13 i=1,np
      INDEX(I,3) = 0
13    continue
      do 14 i=1,np
c      search for pivot element
      qmax = 0
      do 15 j=1,np
      IF(INDEX(J,3).eq.1) go to 15
      do 16 k=1,np
      IF(INDEX(K,3).gt.1) go to 1670
      IF(INDEX(K,3).eq.1) go to 16
      IF(qmax .gt.abs(b(j,k))) goto 16
      IROW = J
      ICOL = K
      qmax = ABS(B(J,K))
16    continue
15    continue
      INDEX(ICOL,3) = INDEX(ICOL,3) + 1
      INDEX(I,1) = IROW
      INDEX(I,2) = ICOL
c      interchange rows to put pivot on diagonal
      IF(iROW.eq.icol) go to 1638
      determ=(-1.0)*determ
      do 17 j=1,np
      br=b(irow,j)
      bc=b(icol,j)
      b(icol,j)=br
      b(irow,j)=bc
17    continue
      rr=rhs(irow)
      rc=rhs(icol)
      rhs(irow)=rc
      rhs(icol)=rr
c      divide pivot row by pivot element
1638   PIVOT = B(ICOL,ICOL)
      DETERM = DETERM * PIVOT
      B(ICOL,ICOL) = 1.
```

```
      do 19 j=1,np
      B(ICOL,J) = B(ICOL,J)/PIVOT
19    continue
      RHS(ICOL) = RHS(ICOL)/PIVOT
c      reduce non-pivot rows
      do 20 j=1,np
      IF(j.eq.icol) goto 20
      T = B(J,ICOL)
      B(J,ICOL) = 0
      bcd(j,icol)=b(j,icol)
      do 21 k=1,np
      B(J,K) = B(J,K) - B(ICOL,K)*T
      bcd(j,k)=b(j,k)
21    continue
      RHS(J) = RHS(J) - RHS(ICOL)*T
20    continue
14    continue
c      interchange columns
      do 22 i=np,1,-1
      IF(INDEX(I,1).eq.index(i,2))go to 22
      IROW = INDEX(I,1)
      ICOL = INDEX(I,2)
      do 23 j=1,np
      br=b(j,irow)
      bc=b(j,icol)
      b(j,irow)=bc
      b(j,icol)=br
23    continue
22    continue
c      test for singularity
      do 24 i=1,np
      IF(index(i,3).ne.1) go to 1670
24    continue
1670   continue
      return
      end
c      ******************************** ***************** ********
c      SUBROUTINE DELTAP
      subroutine deltap
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
      common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xsca1(30),dendi1(30)
      common /zxd/ fcalc(300), ftemp(300),varwt(300,2), bcd(4,4)
      common /zxe/ fcalc2(300)
```

```
      common /main/ nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
c
c      Modifies parameters according to changes in rhs
c      ***************************** ******************* ********
      do 25 i=1,np
      P(I) = P(I) - RHS(I) * FRACT
25    continue
      return
      end
c      ******************************** ***************** *******
c      subroutine pslope
c      2/1/82 by Dave Whitman
c      calculates partial of the function with
c      respect to the each of the parameters at
c      each of the observations, and stores them in dfdp.
c      ******************************** ***************** *******
      subroutine pslope
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
      common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xscal(30),dendil(30)
      common /zxd/ fcalc(300), ftemp(300),varwt(300,2) ,bcd(4,4)
      common /zxe/ fcalc2(300)
      common /main/nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
      do 28 is=1,nobs
      FTEMP(IS) = FCALC(IS)*WEIGHT
28    continue
      do 29 is=1,np
      TP = P(IS)
      P(IS) = TP*1.000005
      if(p(is).eq.0)p(is)=tp+0.000005
      call calc
      DPP=0
      do 30 js=1,nobs
      dvp=0.000005*tp
      if (tp.eq.0.0) dvp=0.000005
      dfdp(js,is)=(fcalc(js)*weight - ftemp(js))/dvp
30    continue
      P(IS) = TP
29    continue
      RETURN
      end
c      ***************************** ******************* *******
```

```
c       SUBROUTINE initialize
      subroutine suinit(ift)
c
c       Prompts for name of input file, then reads problem in
c       ****************************** ******************** ******
c
      implicit integer (g-n)
      implicit double precision (a-f,o-z)
      character*80 ifile
      character*80 ofile
      character*80 ifil1
      character*80 pname
       common p(4),dfdp(300,4), index(4,5),b(4,4), aobs(300)
       common /zxa/ obswt(300), vobs(300,2), v(300,2)
       common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
       common /zxc/ xscal,dendif, xscal(30),dendi1(30)
       common /zxd/ fcalc(300),ftemp( 300),varwt(300,2), bcd(4,4)
       common /zxe/ fcalc2(300)
       common /main/ nvar,np,nft,numit, fract
       common /main2/ devsq1,devsq2, nobs,devsq,weight
      common /chad/ ifile,ifil1(30), pname(4)
       if(ift.gt.1)go to 1809
       write(6,*)' enter output filename'
       read(5,177)ofile
       open(unit=2, file=ofile,status= 'new')
      177 format(a)
       write(6,*)' enter number of data files to be processed'
       read(5,100)nft
100     format(i3)
       do 30 i=1,nft
       write(6,110)i
110     format( '$ name of input file#', 1x,i3,1x,'?')
120     format(a)
       read(5,120)ifil1(i)
       write(6,*) 'enter x-scale and density difference'
       read(5,*) xscal(i),dendi1(i)
       write(1,1012) xscal(i), dendi1(i)
1012    format( ' x-scale = ',f12.8,' density difference =
     ',f12.8)
30      continue
1809     IFILE =IFIL1(ift)
       xscal=xscal(ift)
       dendif=dendi1(ift)
52      open(unit=1, file=ifile, status='old',err=50)
       go to 48
50      continue
      write(6,*)'file does not exist'
```

```
      write (6,110)ift
      read(5,120)ifil1(ift)
      ifile=ifil1(ift)
      close (1)
      go to 52
48     continue
      read(1,180)nobs
180    format(i4)
      read(1,180)numit
      read(1,180)iuserwt
      read(1,180)internalwt
      read(1,182)fract
      do 34 i=1,nobs
      read(1,182)aobs(i)
182    format(f12.7)
      if(iuserwt.eq.1)read(1,182) obswt(i)
      if(iuserwt.ne.1) obswt(i)=1
      do 36 j=1,nvar
      read(1,182)vobs(i,j)
      if(iuserwt.eq.1)read(1,182) varwt(i,j)
      if(iuserwt.ne.1)varwt(i,j)=1
36     continue
34     continue
      do 37 i=1,np
      read(1,183)pname(i)
183    format(a13)
      read(1,182)p(i)
37     continue
      DEVSQ1 = 1D+20
      devsq2 = 1d+20
     close (1)
      return
      end
c      ****************************** ****************** ********
c      SUBROUTINE REPORT
c
c      ****************************** ****************** ********
      subroutine report(ift,nonc,nond)
     implicit integer (g-n)
     implicit double precision (a-f,o-z)
     character*80 ifile
     character*80 ifil1
     character*80 pname
     common p(4),dfdp(300,4), index(4,5),b(4,4),aobs(300)
      common /zxa/ obswt(300),vobs(300,2), v(300,2)
      common /zxb/ dfdv(300,2), iflag(300),dlamda(300), rhs(4)
      common /zxc/ xscal,dendif, xscal(30),dendil(30)
```

```
      common /zxd/ fcalc(300), ftemp(300), varwt(300,2), bcd(4,4)
      common /zxe/ fcalc2(300)
      common /main/nvar,np,nft,numit, fract
      common /main2/ devsq1,devsq2, nobs,devsq,weight
     common /chad/ ifile,ifil1(30), pname(4)
     if(ift.gt.1)go to 1663
140   format( ' DATA FILE: ',2x,a13)
     write(2,140)ifile
      write(2,*)
     write(2,*)'scale, cm/division, = ',xscal
     write(2,*)
     write(2,*)'density difference, g/cm**3, = ',dendif
     write(2,*)
     write(2,*)
     write(2,*)
     write(2,*)
1663  continue
     write(2,*)
     write(2,187)ift
187   format('iteration #',1x,i3)
     if(nonc.ne.1.and. nond.ne.1)go to 1387
      write(2,*) ' Function Values'
      write(2,143)
143    format(6x,'xcalc',11x,'xobs', 10x,'ycalc',11x,'yobs')
      write(2,144)
144   format(6x,'——',11x,'——', 10x,'——',11x,'——')
      if(nonc.eq.1)go to 1330
      if(nond.eq.1)go to 1330
1387  continue
      if(ift.eq.numit) go to 1330
      DEVSQ = 0
      go to 1333
1330  l=1
      do 43 i=1,nobs
     xpd=vobs(i,1)-p(3)
     if(vobs(i,1).lt.0.0)xpd= vobs(i,1)-p(3)
     ypd=aobs(i)-p(4)
      WRITE(2,145)vobs(i,2),xpd, fcalc2(i),ypd
      DEVSQ = DEVSQ + FCALC(I)**2* OBSWT(I)*WEIGHT
43    continue
1333  changg=devsq1-devsq2
      SURT=-979.699*DENDIF*XSCAL**2/ p(2)
     write (2,149) p(1),p(2)
149   format('r0= ',f12.7,5x,'beta= ',f12.7)
     write (2,179) p(3),p(4)
179   format('x0= ',f12.7,5x,'y0= ',f12.7)
      write(2,148) surt
```

```
145   format(3x,f12.7,3x,f12.7,3x,f12.7, 3x,f12.7)
148   format(' surface/interfacial tension = ',f12.2)
      write(2,159)devsq1,changg
159   format(' sum(objective function)**2 = ',e13.6,
      1' delta sum= ',e13.6)
      return
      end
```

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. O. A. Basaran
3-7. C. H. Byers
8. S. D. Clinton
9. P. W. Fisher
10. R. K. Genung
11. M. R. Gibson
12-16. M. T. Harris
17. J. R. Hightower
18. E. K. Johnson
19. F. R. Mynatt
20. J. J. Perona
21. F. M. Scheitlin
22. C. D. Scott
23. T. C. Scott
24. W. G. Sisson
25. M. G. Stewart
26. A. Zucker
27-28. Central Research Library
29-30. Laboratory Records
31. Laboratory Records, ORNL R.C.
32. ORNL Patent Section
33. ORNL-Y-12 Technical Library
Document Reference Section

EXTERNAL DISTRIBUTION

34. F. J. Hurst, 134 Nebraska Ave., Oak Ridge, TN, 37830
35. H. Hsu, Department of Chemical Engineering, Univ. of Tennessee, Knoxville, TN
36. R. S. Marianelli, Office of Basic Energy Sciences, DOE, Room G-338/GTN, Washington, DC 20545
37. F. D. Stevenson, Office of Basic Energy Sciences, DOE, Room G-341/GTN, Washington, DC 20545
38. Office of Assistant Manager for Energy Research and Development, DOE-ORO, P.O. Box 2001, Oak Ridge, TN. 37831-8600
39-48. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831

☆U.S. GOVERNMENT PRINTING OFFICE 1989—748-114/00122