The Institute of Paper Science and Technology

Atlanta, Georgia

Doctor's Dissertation

A Computational Fluid Dynamics Model for Transient Three-Dimensional Free Surface Flows

John Ferney McKibben

November, 1993

A COMPUTATIONAL FLUID DYNAMICS MODEL FOR TRANSIENT THREE-DIMENSIONAL FREE SURFACE FLOWS

A Thesis Submitted by

John Ferney McKibben

B.S. 1984, Oregon State University

B.S. 1985, Southern Oregon State College

M.S. 1987, Lawrence University - The Institute of Paper Chemistry

in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
from the Institute of Paper Science and Technology
Atlanta, Georgia

November, 1993

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Free surface problems occur in a variety of processes important in the manufacture of pulp and paper. Examples include black liquor spraying in a recovery furnace, jets from head boxes, the forming section of a paper machine, condensate flow in dryer cylinders, and finishing operations such as coating and polymer extrusion.

The purpose of this work was to develop a computational fluid dynamics model for the analysis of some of these free surface problems. Specifically, the features added to an available computational technique have allowed the study of the instability of a thin viscous sheet of fluid flowing through an inviscid vapor phase. This problem has direct application in the understanding of black liquor spraying.

In order to accurately solve the sheet instability problem, it was necessary to accurately include the deviatoric normal stress in the liquid phase in the interfacial boundary condition arising from a normal stress balance. The driving force for sheet instability, variations in the vapor phase pressure must also be allowed. In this computational technique, vapor phase pressure variations are determined by solution of potential flow in the vapor phase coupled to the solution of the full Navier-Stokes equations in the liquid phase through both the continuity of normal velocity at the interface and the interfacial normal stress balance.

The accuracy of this computational technique is demonstrated through solution of the lid-driven cavity problem for confined flows, the die-swell problem for free surface flows (with and without surface tension), and the stability of a thin viscous sheet flowing through a stagnant, inviscid vapor phase. Accurate solution of these test problems indicates that the new features of this computational technique work properly. Additional

problems studied include flows under the blade in a short dwell coater and condensate flows in dryer cylinders.

Both temporal and spatial pressure variations can occur in the pond of a short dwell time coating applicator. Since the flow under the blade is made up of a combination of Couette and Poiseuille flows, these pressure variations upstream of the blade effect the flow under the blade and the resulting coating thickness. The effects of both temporal and spatial variations on film thickness are presented.

The heat transfer rate through the condensate layer in a dryer cylinder can limit the paper drying rate. In this work, the film thickness and velocity variations expected within the condensate layer are predicted. The resulting velocity profiles show a small boundary layer accompanied by an "inviscid" core of relatively constant velocity.

In this dissertation, I have developed a computational technique based on the volume of fluid technique for tracking the interface and a modified form of the SOLA solution algorithm for solution of the Navier-Stokes equations. This technique includes third order accurate treatment of the advective terms in the Navier-Stokes equations, the liquid phase deviatoric normal stress at the interface, and allows for pressure variation in the vapor phase through solution of potential flow. This computational technique is unique in its ability to solve the coupled problem of an initially stagnant, inviscid vapor phase governed by potential flow and a moving liquid phase governed by the incompressible Navier-Stokes equations.

INTRODUCTION

Many processing operations in the pulp and paper industry involve flows of fluids with free surfaces. These operations include the spraying of black liquor into a recovery furnace, jets leaving the headbox, the paper machine forming section, condensate flows inside dryer cylinders, coating application systems, and finishing operations such as polymer film extrusion.

In the context of this thesis, free surface flows are defined as flows where at least a portion of a liquid phase of interest is bounded by a vapor phase rather than a solid wall. The presence of a free surface complicates solution of the flow equations by requiring some means of tracking the location of the interface between the liquid and vapor phases in addition to the already difficult task of solving for the pressure and velocity fields within the flow. Specifically, this work focuses on developing a computation tool suitable for studying the problem of spraying black liquor into a recovery furnace. I begin by discussing the importance of black liquor combustion on the economic viability of the Kraft pulping process, followed by a discussion of the desirable characteristics of a black liquor spray.

THE KRAFT PULPING PROCESS

Wood, consisting primarily of cellulosic fibers held together by a "glue" called lignin, is the primary raw material in the manufacture of paper. In order to manufacture paper, individual fibers are required. The fibers can be separated from the lignin matrix by a variety of means, with one of the most common being the Kraft pulping process.

In the Kraft process, wood chips and white liquor, an aqueous solution of sodium sulfide and sodium hydroxide, are heated in a reaction vessel called a digester. The lignin matrix is preferentially attacked by the inorganic pulping chemicals and breaks down into soluble fragments freeing the individual fibers from the lignin matrix.

In order to recover the inorganic pulping chemicals, now primarily in the form of sodium sulfate and sodium carbonate, the fibers are washed and sent on to bleaching or papermaking operations. The filtrate from the washed fibers, termed weak black liquor because of its color, contains the inorganic chemicals that must be recovered and the dissolved lignin fragments.

In order to make the Kraft pulping process economically viable, it is necessary to recover the inorganic chemicals as well as the chemical energy contained in the lignin fragments.[1,2] These objectives are accomplished by combustion of the black liquor in a recovery furnace after concentration in multiple effect evaporators to 65-75% solids. The concentrated black liquor is burned in a water-walled recovery furnace, with the energy released used to produce high pressure steam for electricity generation and to provide process steam for plant operation. The inorganic component of the black liquor is recovered from the bottom of the furnace as a smelt of molten salts consisting primarily of sodium sulfide and sodium carbonate.

The final step in the recovery of the inorganic pulping chemicals is the causticizing step where the sodium hydroxide is renewed. First, the sodium sulfide and sodium carbonate salts are dissolved in water. Next, lime (calcium oxide) is added to the solution precipitating calcium carbonate from a solution now consisting of sodium sulfide and sodium hydroxide. Finally, the calcium carbonate is recovered and heated in a kiln to drive off carbon dioxide yielding calcium oxide.

The general steps in the pulping and chemical recovery cycles are presented in Figure 2. As mentioned above, the economic viability of the Kraft pulping process is determined by the recovery of the pulping chemicals and the chemical energy in the dissolved lignin. Therefore, the operating characteristics of the recovery furnace are of great importance to mill operations.

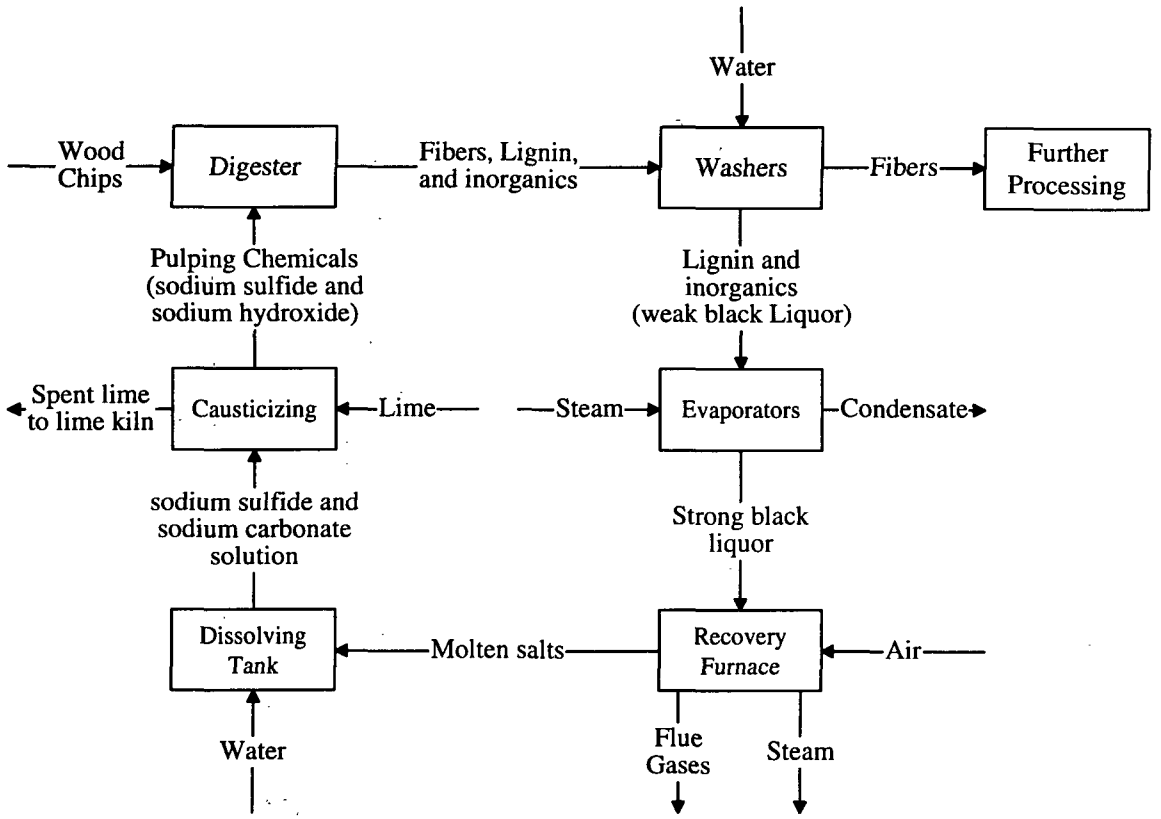

Figure 1. General steps in the pulping and chemical recovery cycles.

CHARACTERISTICS OF A BLACK LIQUOR RECOVERY BOILER

A diagram of a typical black liquor recovery furnace is presented in Figure 2. The recovery boiler is similar to other large industrial furnaces in that fuel is burned in the combustion zone and steam is generated in the heat exchangers. The primary differences

are due to the presence of relatively large amounts of non-combustible inorganic species in black liquor which are generally not present to the same degree in other fuels.

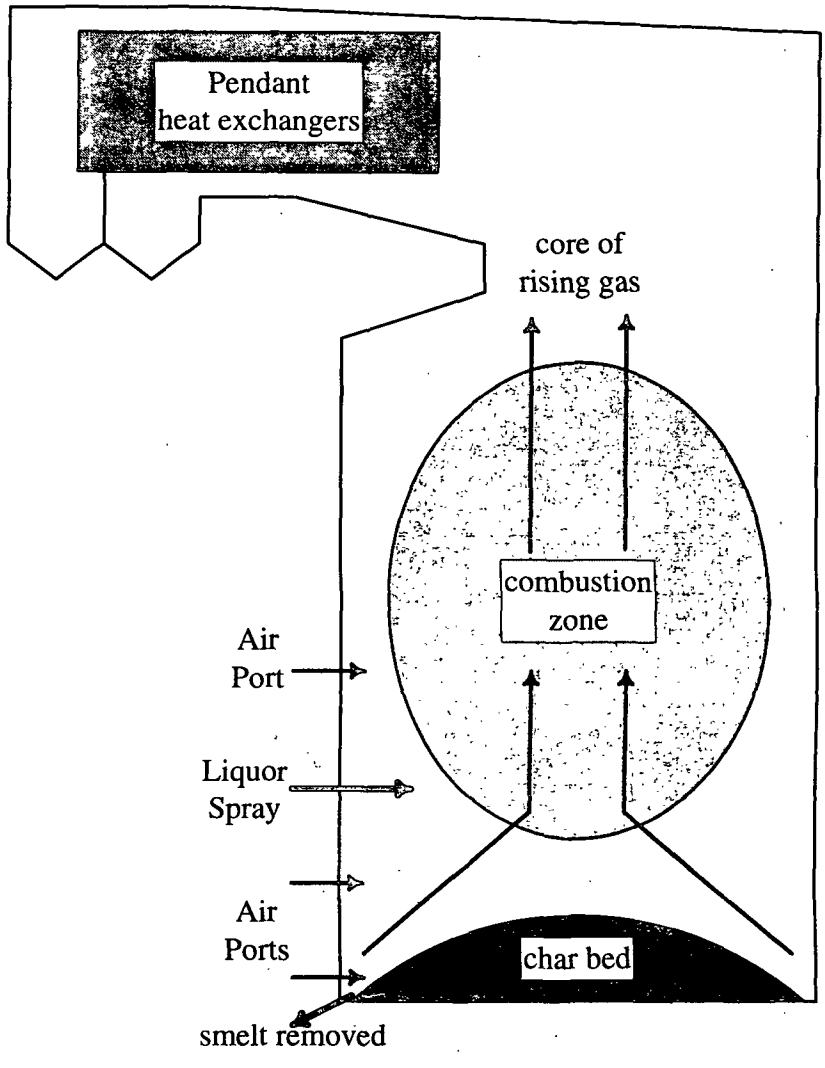

Figure 2. Diagram of a typical black liquor recovery furnace.

The presence of these inorganic species leads to fouling of heat transfer surfaces by molten salts. In addition, the presence of the liquid smelt in the lower furnace is a safety hazard due to the possibility of smelt-water explosions in the event of water tube rupture.[1,2]

In order to understand the implications of black liquor combustion on furnace operation, it is necessary to review the steps in black liquor combustion. These steps actually occur at overlaping times, but are usually discussed as occurring in series. The first step is the drying phase where the remaining water in the droplets is evaporated. This is followed by the gasification stage where combustible gases are given off through pyrolysis. The final stage in the black liquor combustion process is char combustion where residual carbon in the char remaining after pyrolysis is burned and the molten smelt formed.

Ideally, the drying and combustion phases occur in the combustion zone while the char combustion step happens either inflight or on the char bed. In reality, each droplet follows a different trajectory during its time in the furnace based on its size, initial velocity, and the air flow patterns in the furnace. Thus, the actual locations in the furnace where these steps occur varies from drop to drop.

Before discussing the probable fate of droplets with different initial sizes, I would like to discuss the effect of droplet size on the rate of the different processes that the droplet undergoes. Frederick[3] reports that the drying time increases linearly with the initial drop diameter. The time for pyrolysis increases with the initial drop size to the five-thirds power.[3] Finally, the time for inflight char combustion increases with the initial drop diameter to the five-thirds power.[3] Thus, the time for all of these processes increases with the drop diameter and the density of the droplet is expected to initially decrease during the drying phase, drop due to swelling[3,4,5] in the pyrolysis phase, and finally increase during the smelt formation step.

Industrial black liquor sprays are known to produce a distribution of droplet sizes.[6] In many combustion processes, smaller droplets are preferred because the overall

rates of the combustion steps increase with decreasing droplet size. The presence of the relatively large inorganic component of black liquor means that drops which are either too small or too large are undesirable.

A small droplet is expected to dry very rapidly and swell. Because small drops have a higher surface area to volume ratio than large drops,[6] the aerodynamic drag on small droplets is relatively higher than that on large droplets making them more susceptible to entrainment in the upward flowing central core. If the pyrolysis reactions and char combustion phases are completed prior to a small droplet entering the heat transfer regime, the droplet, now consisting of molten smelt, may be expected to drop to the bottom of the furnace since the smelt has a much higher density and may be able to overcome the aerodynamic drag. In the event that the droplet remains in the gas stream, it is expected to contribute to fouling the heat exchangers in the upper part of the furnace, reducing the efficiency of energy recovery.

Thus, we see that droplets that are too small are expected to lead to problems in the heat transfer section due to fouling. Droplets that are too large yield a different set of problems. Efficient regeneration of the sodium sulfide pulping chemical in the lower part of the furnace requires both high temperature and a reducing atmosphere. Droplets that are too large may not dry completely prior to reaching the char bed, resulting in low char bed temperature due to the energy required to evaporate the remaining water. In addition, there are safety concerns, because the addition of water to the smelt bed can lead to an explosive smelt-water interaction.[1,2]

BLACK LIQUOR SPRAYING

Industrially, black liquor is sprayed using either hollow cone swirl, v-jet, or splash-plate type nozzles.[6] All of these nozzle types share the characteristic that a sheet of fluid is formed which subsequently becomes unstable and breaks up into droplets.[6] This is in contrast to jet-producing nozzles where a cylindrical strand of fluid is produced.

In his doctoral dissertation at the Institute of Paper Science and Technology, Spielbauer[6] studied the mechanism of the breakup of a radially thinning sheet into individual droplets. He described the following sequence of events leading to droplet formation. First, the liquid sheet becomes unstable due to aerodynamic forces and sinuous waves in the sheet are observed. Next, the sheet perforates due to an undetermined process, but probably some combination of non-wettable particles, air bubbles, impinging droplets or particles, and local thin spots due to multiple growing wavelengths. Thirdly, the perforations in the sheet grow due to surface tension forces and interact to form a "web" of strands. Finally, the strands become unstable and break up into droplets.

The stochastic nature of the perforation mechanism yields strands of varying sizes, which implies that the resulting droplets will be of various sizes. Thus, a distribution of droplet sizes is expected from nozzles where this mechanism is active, in agreement with experimental results.[6]

The growth of waves in the sheet of black liquor, to a large extent, determines the distribution of droplets obtained from a nozzle. As a result, the primary purpose of this work is to develop a computational tool capable of studying the growth of waves in a thin viscous sheet of fluid flowing through an inviscid vapor phase.

I have discussed the importance of the black liquor recovery cycle and the characteristics of the black liquor spray. One of the most important processes in black liquor spraying is the instability of the sheet of black liquor prior to sheet breakup. This instability leads to the thinning of the sheet which ultimately results in perforations and sheet breakup. This effort to develop a tool suitable for the study of the liquid sheet instability problem involved the following steps:

- A mathematical description of the equations describing fluid flow.
- A mathematical definition of a free surface.
- A review of methods which have previously been used to solve free surface flow problems.
- Additional capabilities needed to study free surface flow problems affecting the pulp and paper industry (beyond those in existing techniques) and a problem statement.
- A detailed description of a solution algorithm (SOLA)[7] for the Navier-Stokes equations coupled with the volume of fluid (VOF)[8] method for tracking the location of the free surface.
- A discussion of the major additions to the SOLA-VOF[8] family of computational techniques needed to solve the free surface problems studied in this work.
- Several validation problems which prove the accuracy and capabilities of different pieces of the IPST-VOF3D code.
- Applications of the IPST-VOF3D computational technique to free surface problems of interest to the paper industry, specifically three-dimensional flows under the blade in a coating application and the flow of condensate inside a dryer cylinder.
- Additional features that would be useful in the computational technique and areas for future work.

DEFINITION OF THE FREE SURFACE PROBLEM

The flow of incompressible, isothermal, Newtonian fluids can be described by the Navier-Stokes equations (NSE) first derived by Navier[9] and Stokes[10]. The first equation,

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

derived from conservation of mass, is commonly referred to as the continuity equation. The second equation,

$$\underset{\text{inertial}}{\frac{\partial \mathbf{u}}{\partial t}} + \underset{\text{advective}}{\mathbf{u} \cdot \nabla \mathbf{u}} = \underset{\text{body forces}}{\mathbf{g}} - \underset{\text{pressure}}{\frac{1}{\rho}\nabla p} + \underset{\text{viscous}}{\nu \nabla^2 \mathbf{u}}, \tag{2}$$

results from a momentum balance and consists of inertial, convective, gravitational or body forces, and a surface force due to the normal stress consisting of pressure and viscous contributions. Here, $\mathbf{u}$ is the velocity vector, $\mathbf{g}$ is the body force vector, p is pressure, ρ is the fluid density, and ν is the fluid kinematic viscosity.

In order to properly pose the problem, appropriate boundary conditions must be specified. These normally include some combination of no-slip walls, symmetry planes (slip walls), continuative outlets, and periodic outlet conditions. Table 1 shows the mathematical descriptions for these boundary conditions on the left edge of a computational domain (smallest value in the x-direction). Analogous conditions may be written for the remaining domain boundaries. Generally, interior obstacles are treated either as slip or no-slip walls with the appropriate conditions applied in a manner similar to the external boundaries. Numerically, boundary conditions for the pressure adjacent to no-slip walls are required and are typically derived from the velocity boundary conditions and the NSE. The specifics of the pressure boundary condition derivation for this computational technique are presented below along with the discussion of the implementation of the computational technique.

Table 1. Boundary Conditions for confined flows.

Boundary Condition	u	v	w	p
Symmetry Plane or Frictionless Wall	$u = 0$	$\frac{\partial v}{\partial x} = 0$	$\frac{\partial w}{\partial x} = 0$	$\frac{\partial p}{\partial x} = 0$
No-Slip Wall	$u = 0$	$v = 0$	$w = 0$	—
Continuative Outlet	$\frac{\partial u}{\partial x} = 0$	$\frac{\partial v}{\partial x} = 0$	$\frac{\partial w}{\partial x} = 0$	$\frac{\partial p}{\partial x} = 0$
Periodic Condition	$u_L = u_R$	$v_L = v_R$	$w_L = w_R$	$p_L = p_R$

The presence of a free-surface yields an interface between two fluids of different properties where the boundary is allowed to move as a function of time. Thus, the addition of a free surface requires additional boundary conditions at the interface between the two phases. These conditions for two general fluids, derived from conservation of normal velocity, normal stress, and tangential stress, respectively, are[11]

$$\mathbf{u}_\ell \cdot \mathbf{n} = \mathbf{u}_v \cdot \mathbf{n}, \tag{3}$$

$$p_\ell - \mathbf{n} \cdot \boldsymbol{\tau}_\ell \cdot \mathbf{n} = p_v - \mathbf{n} \cdot \boldsymbol{\tau}_v \cdot \mathbf{n} + \sigma \kappa, \tag{4}$$

$$\mathbf{n} \cdot \boldsymbol{\tau}_\ell \cdot \mathbf{s} = \mathbf{n} \cdot \boldsymbol{\tau}_v \cdot \mathbf{s}, \tag{5}$$

and $$\mathbf{n} \cdot \boldsymbol{\tau}_\ell \cdot \mathbf{t} = \mathbf{n} \cdot \boldsymbol{\tau}_v \cdot \mathbf{t} \tag{6}$$

where the subscripts ℓ and v refer to the liquid and vapor phases, respectively; $\mathbf{n}$ is the unit vector normal to the interface; $\mathbf{s}$ and $\mathbf{t}$ are unit vectors tangent to the interface that form an orthogonal coordinate system with $\mathbf{n}$; σ is the surface tension between the fluids; κ is the curvature of the interface; and τ is the deviatoric stress tensor (the viscous component of the total stress). For a Newtonian fluid, the deviatoric stress tensor is defined as

$$\tau = \mu\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^{T}\right], \tag{7}$$

where μ is the Newtonian viscosity.[11]

The primary assumption used in deriving a free surface boundary condition from the boundary conditions for a general interface just presented is that the vapor phase is inviscid. This yields

$$p_{\ell} - \mathbf{n} \cdot \tau_{\ell} \cdot \mathbf{n} = p_{v} + \sigma \kappa \tag{8}$$

$$\mathbf{n} \cdot \tau_{\ell} \cdot \mathbf{s} = 0 \tag{9}$$

and $$\mathbf{n} \cdot \tau_{\ell} \cdot \mathbf{t} = 0 \tag{10}$$

for the normal and tangential stress conditions at the interface with the normal velocity condition remaining the same. Usually, the additional assumption of constant pressure in the vapor phase is also made. This assumption simplifies the problem considerably by eliminating the need to solve the NSE in the vapor phase. Further discussion of the implications of the constant vapor phase pressure assumption are presented in the literature review section below.

If the deviatoric stress in the liquid phase is also assumed to be small, the tangential conditions vanish and the condition arising from the normal stress balance is represented by Laplace's formula,

$$p_{\ell} = p_{0} + \sigma \kappa, \tag{11}$$

where p_0 is the constant vapor phase pressure.

The unit vector normal to the interface, two mutually orthogonal unit vectors tangential to the interface, and interfacial curvature can be defined in terms of the local height of the interface above a plane,

$$H(x,y,z) \equiv y - \eta(x,z) = 0, \tag{12}$$

where η is an auxiliary height function. The vector normal to the interface can be

computed as the gradient of the H(x, y, z).[12] For Cartesian coordinates the normal vector becomes

$$\nabla H(x,y,z) = \frac{\partial(y-\eta(x,z))}{\partial x}\,\mathbf{i} + \frac{\partial(y-\eta(x,z))}{\partial y}\,\mathbf{j} + \frac{\partial(y-\eta(x,z))}{\partial z}\,\mathbf{k} \qquad (13)$$
$$= -\eta_x\,\mathbf{i} + \mathbf{j} - \eta_z\,\mathbf{k}$$

where **i**, **j**, and **k** are the unit vectors in the coordinate directions, and subscripts denote partial differentiation. To generate the unit normal, this vector must be scaled to unit length yielding the unit normal vector,

$$\mathbf{n} = \frac{-\eta_x\,\mathbf{i} + \mathbf{j} - \eta_z\,\mathbf{k}}{\sqrt{\eta_x^2 + 1 + \eta_z^2}}. \qquad (14a)$$

The tangential vectors are derived from mutual orthogonality of the normal and tangential vectors condition and by setting the i-component of **s** tangential vector to zero, yielding

$$\mathbf{s} = \frac{\eta_z\,\mathbf{j} + \mathbf{k}}{\sqrt{1+\eta_z^2}} \qquad (14b)$$

and $$\mathbf{t} = \mathbf{n} \times \mathbf{s} = \frac{\left(1+\eta_z^2\right)\mathbf{i} + \eta_x\,\mathbf{j} - \eta_x\eta_z\,\mathbf{k}}{\sqrt{1+\eta_z^2}\sqrt{\eta_x^2+1+\eta_z^2}} \qquad (14c)$$

for the unit tangential vectors.

Finally, the surface curvature is defined as the negative divergence of the unit normal vector[13]

$$\kappa = -\nabla \cdot \mathbf{n}$$
$$= -\frac{\partial}{\partial x}\left[\frac{-\eta_x}{\sqrt{\eta_x^2+1+\eta_z^2}}\right] - \frac{\partial}{\partial y}\left[\frac{1}{\sqrt{\eta_x^2+1+\eta_z^2}}\right] - \frac{\partial}{\partial z}\left[\frac{-\eta_z}{\sqrt{\eta_x^2+1+\eta_z^2}}\right] \qquad (15)$$

which simplifies to

$$\kappa = \frac{\eta_{xx}\left(1+\eta_z^2\right)+\eta_{zz}\left(\eta_x^2+1\right)}{\left(\eta_x^2+1+\eta_z^2\right)^{3/2}}. \tag{16}$$

These formulas for Cartesian coordinates can be rotated to apply to interfaces oriented primarily in the x or z directions.

In cylindrical coordinates, the formulas vary depending on the orientation of the interface. The formulas for each direction again begin with local height functions such as

$$R(r,\theta,z) = r - \eta(\theta,z). \tag{17}$$

The unit normal and tangential vectors are derived using the gradient operator, here for cylindrical coordinates, in the same manner as that used above

$$\mathbf{n} = \frac{\eta\,\mathbf{r} - \eta_\theta\,\theta - \eta\eta_z\,\mathbf{z}}{\sqrt{\eta^2+\eta_\theta^2+\eta^2\eta_z^2}}, \tag{18a}$$

$$\mathbf{s} = \frac{\eta_z\,\mathbf{r}+\mathbf{z}}{\sqrt{1+\eta_z^2}}, \tag{18b}$$

and $$\mathbf{t} = \frac{-\eta_\theta\,\mathbf{r}+\eta\left(1+\eta_z^2\right)\theta+\eta_\theta\eta_z\,\mathbf{z}}{\sqrt{1+\eta_z^2}\sqrt{\eta^2+\eta_\theta^2+\eta^2\eta_z^2}}. \tag{18c}$$

Similarly, for the Θ-direction the local fluid height function is

$$\Theta(r,\theta,z) = \theta - \eta(r,z) \tag{19}$$

with the unit normal and tangential vectors

$$\mathbf{n} = \frac{-r\eta_r\,\mathbf{r}+\theta-r\eta_z\,\mathbf{z}}{\sqrt{r^2\eta_r^2+1+r^2\eta_z^2}}, \tag{20a}$$

$$\mathbf{s} = \frac{r\eta_z\,\theta+\mathbf{z}}{\sqrt{1+r^2\eta_z^2}}, \tag{20b}$$

and $$\mathbf{t} = \frac{-\left(1+r^2\eta_z^2\right)\mathbf{r}-r\eta_r\,\theta+r^2\eta_z\eta_r\,\mathbf{z}}{\sqrt{1+r^2\eta_z^2}\sqrt{r^2\eta_r^2+1+r^2\eta_z^2}}. \tag{20c}$$

Finally, for the z-direction the local height function is

$$Z(r,\theta,z) = z - \eta(r,\theta), \tag{21}$$

with the unit normal and tangential vectors

$$\mathbf{n} = \frac{-\eta_r\,\mathbf{r} - \eta_\theta\,\boldsymbol{\theta} + r\,\mathbf{z}}{\sqrt{r^2\eta_r^2 + \eta_\theta^2 + r^2}}, \tag{22a}$$

$$\mathbf{s} = \frac{r\,\boldsymbol{\theta} + \eta_\theta\,\mathbf{z}}{\sqrt{\eta_\theta^2 + r^2}}, \tag{22b}$$

and

$$\mathbf{t} = \frac{-\left(\eta_\theta^2 + r^2\right)\mathbf{r} + r\eta_r\eta_\theta\,\boldsymbol{\theta} - r^2\eta_r\,\mathbf{z}}{\sqrt{\eta_\theta^2 + r^2}\sqrt{r^2\eta_r^2 + \eta_\theta^2 + r^2}}. \tag{22c}$$

With the definition of a free surface problem complete, I present a summary of the primary difficulties in solving free surface problems numerically, quoted from Floryan and Rasmussen.[14]

1. The interfacial boundary conditions are nonlinear and of a mixed type, and they involve pressure which has to be evaluated accurately at the boundary.

2. The field equations are nonlinear and boundary layers are possible.

3. The solution domain has an irregular, constantly changing geometry; its connectivity may change, e.g. breakup of a liquid droplet.

4. The interface may undergo large distortions and non-analytic cusp-like interfaces are possible.

5. Crossing interfaces may occur when multiple interfaces are involved.

6. Tracking the shape of the interface, i.e., its curvature, demands high accuracy to account properly for the surface tension effects.

7. Presence of singularities at the contact points poses serious difficulties for accurate determination of the location of the boundary.

8. Several physical instabilities are known to occur at the interfaces, and, therefore, several intrinsic temporal and spatial characteristic scales might be involved. Knowledge of these scales is required in order to establish the appropriate numerical step sizes.

This list of potential pitfalls in solving free surface problems implies that no single numerical approach can be expected to accurately and efficiently treat all free surface problems. Each approach may be expected to handle certain classes of problems well, while potentially having great difficulty with other classes of problems.

LITERATURE REVIEW

The presence of a free surface and the accompanying need to track the location of the moving interface further complicates the already difficult task of solving the Navier-Stokes equations. Detailed knowledge of the interface shape and location is required to accurately impose the highly non-linear interfacial boundary conditions presented above. In addition, the location and shape of the interface are often among the most important pieces of information obtained from a free surface problem solution.

This section presents a review of the literature describing techniques for solving free surface problems. This is followed by a discussion of the stability of a thin viscous sheet flowing through an inviscid vapor phase, the problem related to black liquor spraying that has driven the development of this computational technique.

NUMERICAL TECHNIQUES FOR FREE SURFACE PROBLEMS

This review of numerical techniques for treating free surface flows will generally follow the format of Floryan and Rasmussen,[14,15] who present more complete reviews of moving boundary methods. Additional reviews related to this subject are presented by Yeung,[16] Hyman,[17] Laskey et al.,[18] Crank,[19] Bulgarelli et al.,[20] Harlow,[21] and Tseng et al.[22]

Current techniques for computational analysis of free surface flows can be divided primarily into Lagrangian and Eulerian approaches. I will describe and discuss many of the available techniques in each of these categories plus some hybrid techniques that do not fit easily into either the Eulerian or Lagrangian classifications. Next, I will discuss in greater detail the particular family of volume tracking Eulerian techniques used in this

work which have advantages in a broad class of free surface flow problems involving large surface deformations.

Lagrangian Approaches

Lagrangian approaches are generally defined as those where the computational mesh used in the numerical approximation of the NSE is allowed to move with the flow. This simplifies the imposition of the interfacial boundary conditions since the interface lies along an edge of the computational domain. Unfortunately, Lagrangian techniques have the disadvantage that, for flows with large surface deformation or locally large shear rates, the local mesh can become distorted, leading to loss of numerical accuracy and, potentially, numerical stability.[14] In some cases, the grid distortion can become so severe that the mesh becomes entangled leading to failure of the numerical technique.

An example of a purely Lagrangian approach can be found in the LINC (Lagrangian incompressible) method presented by Hirt et al.[23] and extended by Butler.[24] In this technique, the vector quantities (position, velocity, and body accelerations) are stored at the computational cell vertices and the scalar quantities (pressure and stress tensor) at the cell centers. A Poisson equation for the pressure in a cell is derived from the requirement of constant volume in a cell as a function of time. This technique generally suffers severely from the grid distortion problems outlined above.

In an effort to overcome the grid distortion problem, free Lagrangian approaches have been developed.[25,26,27,28] In the free Lagrangian approach, the computational grid is reconstructed at each time step by choosing the nearest neighbors to each vertex. Thus, the advantages of the Lagrangian representation are maintained, while the computation grid is dynamically adjusted to prevent entanglement and maintain accuracy. The usual

approach is again to have the vector quantities vertex centered and the scalar quantities cell centered,[25,26,27,28] but examples exist where all quantities are cell centered[29,30,31] and where all quantities are vertex centered.[32,33]

An alternative to the free Lagrangian approach for removing the grid distortion problem is to allow periodic rezoning of the computational grid. This process maintains the integrity of the computational grid while keeping the advantage of allowing the interface to be represented by an edge of the computational domain. The rezoning process has a side effect of introducing numerical diffusion as the information is transferred between the computational grids. Examples of numerical techniques using rezoning techniques are presented by Hirt et al.,[34] Amsden et al.,[35] Addesio et al.,[36] and Bach and Hassager.[37] As an example, the Arbitrary Lagrangian-Eulerian (ALE) technique developed by Hirt et al.[34] will be discussed in greater detail below as a hybrid technique.

Eulerian Methods

In the Eulerian approach, the computational mesh typically remains fixed or is allowed to move in a prescribed manner. What makes these techniques an Eulerian representation is that, in all of these methods, the fluid moves relative to the mesh. The location of the interface is maintained either by some means of tracking the interface location[38,39,40,41] or by tracking the location of the fluid itself, referred to as volume tracking.[8,42] It is possible to use any of the interface tracking techniques with both the fixed and variable grid formulations.

Fixed Grid Eulerian Methods

Fixed grid representations have the advantage that there is no possibility of the mesh becoming entangled and thus large changes in the free surface can be tolerated. The primary disadvantage with fixed grids versus Lagrangian and variable grid approaches is a loss of precision in the knowledge of the interface location and shape. Interface tracking in fixed grid methods are reviewed by Hyman[17] and Laskey[18] with additional discussion presented by Hirt and Nichols.[8] The two primary examples of interface tracking methods are the height function and line segment approaches.

Height functions are probably the simplest methods for tracking a moving interface. The interface location is stored as the height of the interface above an arbitrary line (typically a coordinate axis) in two-dimensions or plane in three-dimensions. In this instance, the local height functions used to compute the surface curvature discussed above are the same as the interface tracking height functions.

As an example, if the interface remains relatively parallel to the x-z plane in three-dimensions, the height function is expected to be a function of position and time such as $H = f(x,z,t)$. Problems can occur when the slope of the interface, $\partial h/\partial x$ or $\partial h/\partial z$, is greater than the local mesh aspect ratio, $\delta y/\delta x$ or $\delta y/\delta z$.[8] This technique is extremely efficient in terms of storage and computations, but has problems with sharp gradients and is limited to interfaces that are single valued. Examples of fixed Eulerian mesh computational techniques using height functions include SOLA-SURF[7] where the height function is stored at the center of a column of computational cells and an approach developed by Hill[40,41] where the height function is stored at the computational cell edges.

Movement of the interface is governed by the kinematic condition representing the requirement that the interface must move with the fluid,

$$\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + w\frac{\partial h}{\partial z} = v. \tag{23}$$

The line segment approach overcomes the requirement that the interface remain relatively flat and single valued by treating the interface as a collection of points connected by line segments. For accuracy, the distance between the points must be less than the local grid spacing.[8] Each point moves with the local fluid velocity maintaining its position on the interface. In a sense, the line segments move in a Lagrangian manner through the fixed Eulerian mesh. To maintain accuracy and computational efficiency of the simulation, line segments can be added and deleted as necessary. Examples of fixed Eulerian mesh methods based on the line segment approach are given by Nichols and Hirt.[39,43]

The line segment method has difficulty in situations where two interfaces intersect or when an interface folds over on itself (e.g. wave-breaking).[8] These difficulties are compounded when the line segment technique is extended to three-dimensions with additional problems arising when segments need to be added or deleted.

Volume tracking methods originated with the Marker and Cell (MAC) method, the first generally successful free surface computational technique. This technique was originally presented by Harlow and Welch[42,44] and modified by Viecelli,[45] among others. In the MAC method, the location of the fluid within a fixed mesh is tracked by a set of massless marker particles which are moved at the end of each time step in a Lagrangian manner. The interface is assumed to lie somewhere within a cell containing marker

particles and having an empty neighbor. Taking the limit as the number of marker particles becomes infinite, it is possible to track the "fullness" of each computational cell.

The computational cell fullness concept leads to a number of volume tracking techniques which differ primarily in the method used to reconstruct the interface from the "fullness" data. Techniques include Simple Line Interface Calculation (SLIC),[46] a modification to SLIC,[47] a method proposed by Chorin[48] to fit an osculating circle with the same properties as the local fluid configuration, and the Volume of Fluid (VOF) concept.[8,49] These finite difference techniques address the issue of how to take the fullness of a computational cell and its neighbors and convert it into an accurate representation of the interface location and local curvature as demonstrated in Figure 3. The early volume tracking codes were all based on finite difference methods, however, recent applications of the cell fullness concept have included finite element analyses.[50,51,52,53]

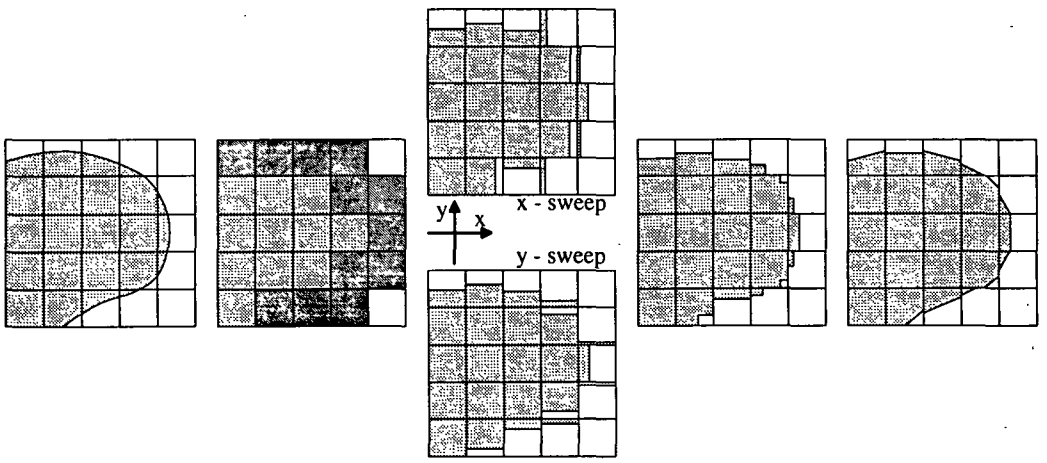

Figure 3. Reconstruction of the interface using various volume tracking procedures from left to right: (a) the actual form of the interface; (b) reconstruction based on a marker-and-cell procedure (the interface is somewhere in the lined area); (c) SLIC reconstruction;[46] (d) improved SLIC reconstruction;[47] (e) VOF reconstruction.[8]

A recent modification to the VOF approach which yields a more accurate representation of the surface tension component of the interfacial boundary condition has been developed.[13,54,55] In this approach, termed the Continuum Surface Force (CSF), the surface tension force is spread over a region near the interface with dimensions on the order of the local computational grid spacing. The resulting force is then treated as an additional body force in the solution of the Navier-Stokes equations. A similar technique for multi-fluid flows has been developed by Unverdi and Tryggvason.[56,57]

The majority of the interface tracking methods and all of the volume tracking methods apply Laplace's formula (11) as the boundary condition at the interface. This is a reasonable assumption for many flows, but the deviatoric normal stress is important in certain problems such as die-swell[40,41] and thin film instability.[58]

Adaptive Grid Eulerian Methods

In adaptive grid Eulerian techniques, after a new surface configuration has been determined a new computational mesh is generated to cover the resulting computational domain. This allows accuracy in applying the interfacial conditions along a domain boundary while eliminating the problems in regions of high shear associated with the Lagrangian methods discussed above. Unfortunately, the problems of mesh entanglement and surface intersection remain. Reviews of adaptive grid methods can be found in Tanner,[59] Denn,[60] Kistler,[61] and Kheshgi.[62]

While most of the fixed grid techniques discussed above use the finite difference method, adaptive grid techniques have been developed which use finite difference methods,[63,64,65] finite element methods,[66,67,68,69,70,71] and spectral element methods.[72]

Mixed Lagrangian-Eulerian Methods

Several techniques have been developed which are hybrids containing features of both Eulerian and Lagrangian methods. These include the ALE method mentioned previously,[34] the Particle in Cell (PIC) method,[73] and the Coupled-Eulerian-Lagrangian (CEL) method.[74] Typically, these methods attempt to maintain the best features of the Lagrangian approach coupled with the best features of the Eulerian approach.

In the ALE method[34] the NSE are solved using the Lagrangian approach, but rezoning is accomplished by convecting the computational nodes upstream at the end of each time step. The level of rezoning can vary from none, yielding a purely Lagrangian representation, to complete rezoning, effectively yielding an Eulerian representation. The ALE approach allows accurate knowledge of the surface location in the Lagrangian frame while reducing the potential for mesh tangling which is a problem with Lagrangian methods. This is accomplished at the expense of a significant amount of numerical diffusion† introduced by the rezoning process.[14]

The PIC[73] method uses Lagrangian particles to transport mass between cells in a fixed Eulerian mesh. The pressure in each cell is determined through an equation of state using the internal energy of the cell and is mass density of the particles.

The CEL[74] method uses a fixed Eulerian mesh to store the velocity and pressure fields. The fluid is tracked using a separate Lagrangian representation. Difficulties exist

---

† Numerical diffusion occurs when the error in a numerical approximation, in this case a first order accurate finite difference scheme, contains second derivatives. The error in the numerical approximation is superimposed on the physical diffusion term exaggerating the magnitude of the diffusion.

in the program logic coupling the Eulerian and Lagrangian frames and this technique has only been implemented for compressible inviscid flows.[14]

Contact Between a Solid and Two Fluid Phases

Thus far, I have discussed the equations to be solved and a number of methods for tracking the interface between the two phases. The remaining problem to discuss is the contact between two-fluid interface an a solid surface, termed the contact point (or line in three-dimensions) depicted in Figure 4. Several theoretical analyses of this problem have been conducted,[75,76,77,78] but the question of the best way to numerically treat the contact point remains open.

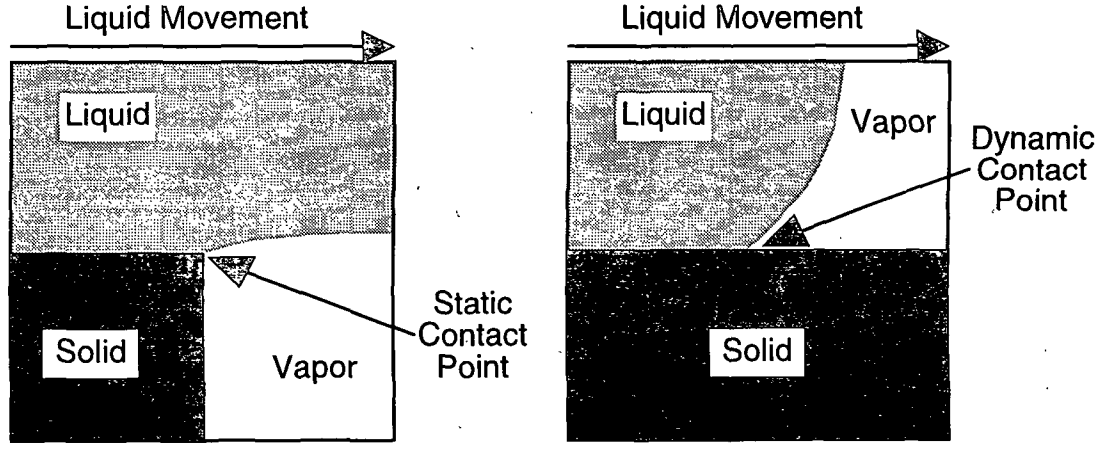

Figure 4. Description of static and dynamic contact points.

The primary difficulty results from a singularity in the stress at the contact point or line.[79] The typical method of treating this singularity is to allow slip in the region of the dynamic contact line. This method, used by Torrey et al.[80,81] and explained in greater detail below, remains unchanged in the current computational technique. For static

contact points, the no-slip condition can be applied and this option has been added to the present computational technique as is discussed below.

The interfacial normal stress balance also requires modification in the region of a contact point. Using Laplace's formula, $p_\ell = p_0 + \sigma \kappa$, as an example, the discontinuity due to the surface tension is modified by an "adhesion" force between the fluid and the wall. This requires a modified treatment of the normal stress discontinuity adjacent to the contact point as is described below.

THE STABILITY OF A THIN VISCOUS SHEET

One of the stages in the black liquor spraying process is the generation of droplets due to the breakup of a thin viscous sheet of black liquor flowing through air. In an effort to develop a tool suitable for studying this process, I have added the capability to solve for potential flow in a vapor phase adjacent to a liquid phase where the full NSE are solved. To test this capability, below I will study the problem of a thin viscous sheet of fluid flowing through a stagnant inviscid vapor phase.

This problem has been studied analytically by many workers with the most complete analysis presented by Li and Tankin.[58] Their analysis begins with a flat sheet moving through a stagnant vapor phase as shown in Figure 5. Superimposed on the sheet is a sinusoidal disturbance of the form

$$\varepsilon = \varepsilon_0 e^{\omega t + ikx} \tag{24}$$

where ε_0 is the initial disturbance amplitude, $\omega = \omega_r + i\omega_i$ is the complex growth rate, $i = \sqrt{-1}$, and k is the wavenumber of the disturbance.

As the fluid with a sinusoidal disturbance moves through the stagnant vapor phase, a difference in pressure builds up along the surface between the crests of the waves and the troughs. At the crest, the relative velocity between the liquid and the vapor is high and, from a simplified analysis using Bernoulli's principle, the pressure in the vapor phase is expected to be low, while at the troughs, the relative velocity is low and the pressure is expected to be high. Thus, the interface is pushed inward toward the fluid in the troughs and expands outward into the vapor at the crests. For low wavenumbers where waves are long, the pressure gradient is relatively small because the low and high pressure regions are relatively far apart, but for high wavenumbers this gradient is high and the growth rate due to the vapor phase pressure is larger.

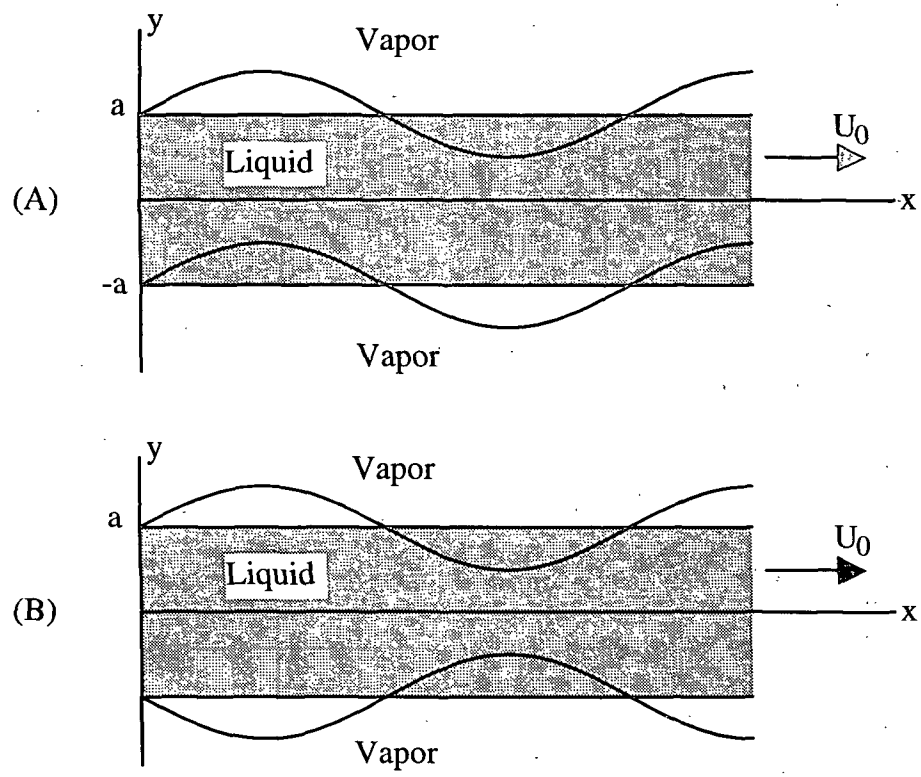

Figure 5. Schematic of the sheet instability problem for antisymmetric, (A), and axisymmetric, (B), disturbances.

Surface tension provides a competing force seeking to maintain a perfectly flat sheet which minimizes the surface energy. At small wavenumbers, the surface curvature is small and, thus, the surface tension force is weak. As the wave number is increased and the curvature increases, the effect of the surface tension becomes more pronounced. This competes with the force due to the pressure gradient in the vapor phase where the magnitude of the gradient is expected to increase with the wave number.

In summary, at low wavenumbers the surface tension restoring force is small and the pressure gradient in the vapor phase is also small, while at high wavenumbers the magnitude of each of these forces is expected to increase. The relationships between the magnitude of these forces and wavenumber are nonlinear and both forces become stronger at increasing wavenumber.

Li and Tankin[58] conducted a linear stability analysis of the problem outlined in Figure 5. That is, the vapor phase was assumed to be inviscid, the liquid phase was assumed initially to have flat surfaces and be moving at uniform velocity, and a disturbance such as that in Equation (24) was imposed. Their analysis yields dispersion relations between the wavenumber of a disturbance to its complex growth rate. In non-dimensional form these relations are

$$0 = \left(\tilde{\omega}_1 + 4m^2Z\right)\tilde{\omega}_1 \tanh(m) + 4m^3Z^2\left[m\tanh(m) + \left(m^2 + \tilde{\omega}_1/Z\right)^{1/2}\tanh\left(\left(m^2 + \tilde{\omega}_1/Z\right)^{1/2}\right)\right] + \tilde{\rho}\tilde{\omega}^2 + m^3 \tag{25}$$

and

$$0 = \left(\tilde{\omega}_1 + 4m^2Z\right)\tilde{\omega}_1 \coth(m) + 4m^3Z^2\left[m\coth(m) + \left(m^2 + \tilde{\omega}_1/Z\right)^{1/2}\coth\left(\left(m^2 + \tilde{\omega}_1/Z\right)^{1/2}\right)\right] + \tilde{\rho}\tilde{\omega}^2 + m^3 \tag{26}$$

for antisymmetric and axisymmetric disturbances, respectively. Here, $\tilde{\omega} = \tilde{\omega}_r + i\,We_\ell^{1/2}\tilde{\omega}_i$; $\tilde{\omega}_1 = \tilde{\omega} + i\,We_\ell^{1/2}m$; $\tilde{\omega}_r = \omega_r\left(\sigma/\rho_\ell a^3\right)^{-1/2}$; $\tilde{\omega}_i = \omega_i(a/U_0)m$; a is the initial sheet half-

thickness; m = ka is the dimensionless wavenumber; and U_0 is the initial sheet velocity. The remaining parameters are the liquid phase Weber number, $We_\ell = \rho_\ell U_0^2 a/\sigma$; the Ohnesorge number, $Z = \mu_\ell(\rho_\ell a\sigma)^{-1/2}$; and the density ratio, $\tilde{\rho} = \rho_v/\rho_\ell$.

It is possible to solve the dispersion relations, (25) and (26), for a given We_ℓ, Z, and $\tilde{\rho}$ to yield the complex growth rate, $\tilde{\omega}$, as a function of the wave number. The computational technique and computer program used to solve this highly nonlinear equation in complex numbers are presented in Appendix VII. The real part of $\tilde{\omega}$ is dimensionless growth rate of a disturbance with wavenumber m. Results from the dispersion relations with $We_\ell = 40$, $Z = 0.1$, and $\tilde{\rho} = 0.1$ are shown in Figure 6 for both antisymmetric and axisymmetric disturbances represented by solid and dashed lines, respectively.

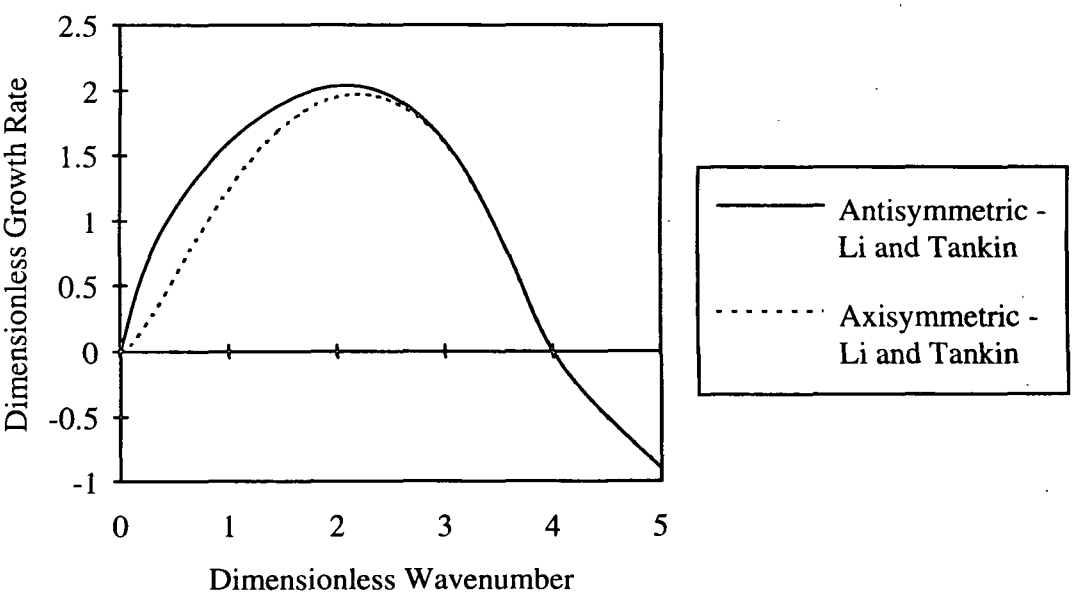

Figure 6. Non-dimensional growth rate for $We_\ell = 40$, $Z = 0.1$, and $\tilde{\rho} = 0.1$ obtained from numerical solution of the Li and Tankin's[58] dispersion relations.

Solutions of modified forms of Li and Tankin's dispersion relations illustrate the importance of the deviatoric normal stress and surface tension components of the interfacial boundary condition. In the absence of surface tension, the dispersion relation for antisymmetric disturbances becomes

$$0=\left(\tilde{\omega}_1+4m^2Z\right)\tilde{\omega}_1\tanh(m) +4m^3Z^2\left[m\tanh(m)+\left(m^2+\tilde{\omega}_1/Z\right)^{1/2}\tanh\left(\left(m^2+\tilde{\omega}_1/Z\right)^{1/2}\right)\right]+\tilde{\rho}\tilde{\omega}^2. \tag{27}$$

Similarly, with the assumption of constant pressure in the vapor phase, the dispersion relation reduces to

$$0=\left(\tilde{\omega}_1+4m^2Z\right)\tilde{\omega}_1\tanh(m) +4m^3Z^2\left[m\tanh(m)+\left(m^2+\tilde{\omega}_1/Z\right)^{1/2}\tanh\left(\left(m^2+\tilde{\omega}_1/Z\right)^{1/2}\right)\right]+m^3. \tag{28}$$

Finally, if Laplace's formula is used as the interfacial boundary condition, neglecting the liquid phase deviatoric normal stress in the interfacial boundary condition, the dispersion relation becomes

$$0=\tilde{\omega}_1^2\tanh(m)+\tilde{\rho}\tilde{\omega}^2+m^3 \tag{29}$$

which readily reduces to the dispersion relation for two inviscid fluid derived by Squire[82] and Hagerty and Shea[83]

$$\tilde{\omega}_r=\frac{m\sqrt{\tilde{\rho}We_\ell\tanh(m)-m\left[\tilde{\rho}+\tanh(m)\right]}}{\tilde{\rho}+\tanh(m)}. \tag{30}$$

Results from the solution of the modified dispersion relations presented in (26), (27), and (30) are compared with results from the complete dispersion relation for antisymmetric disturbances in Figure 7.

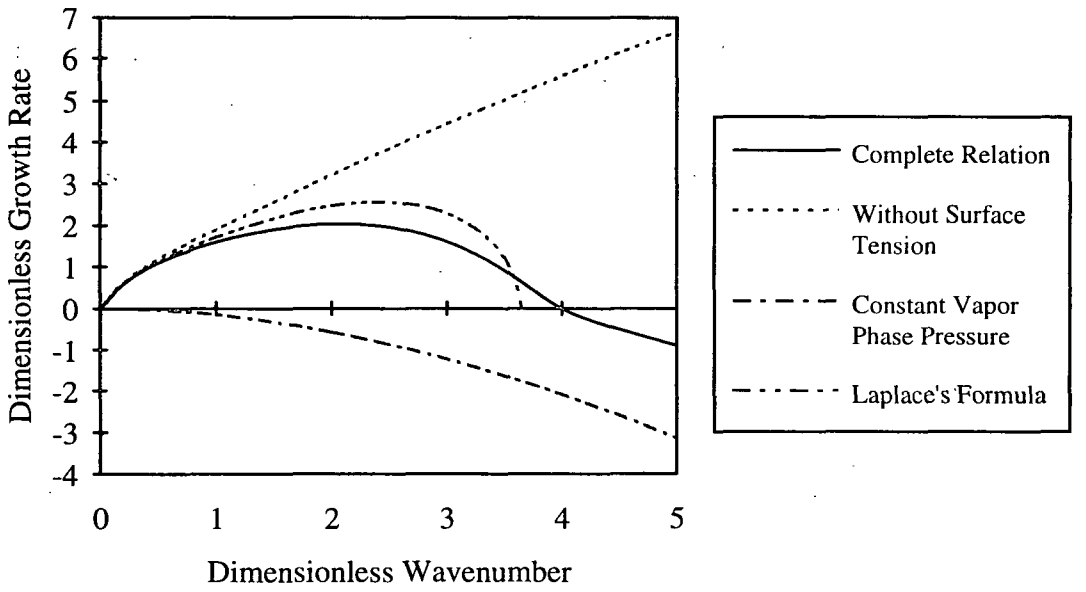

Figure 7. Non-dimensional growth rate for $We_\ell = 40$, $Z = 0.1$, and $\tilde{\rho} = 0.1$ obtained from numerical solution of the Li and Tankin's[58] dispersion relations and modified forms of the dispersion relation.

Significant deviations from the results obtained using the complete dispersion relation are seen when any of the assumptions leading to the modified dispersion relations are made. This means that it is important to include the liquid phase deviatoric normal stress at the interface, the pressure discontinuity at the interface due to surface tension, and the pressure fluctuations in the vapor phase to accurately predict the growth rate of a wave in a viscous liquid sheet flowing through an inviscid vapor phase.

PROBLEM ANALYSIS AND OBJECTIVES

The primary goal of this thesis is to develop a computational technique for analysis of three-dimensional free surface flow problems. Study of the stability of a thin viscous sheet flowing through a stagnant vapor phase, a phenomenon associated with sheet breakup in black liquor spraying, has been the driving force for this work.

Many modeling techniques have been applied to free surface problems as outlined above; however, all of the previous analyses suffer from limitations preventing their direct application to the sheet instability problem. The large deformations and potentially discontinuous interfaces at sheet breakup suggest that the volume tracking techniques are the most appropriate for this problem due to their simplicity in treating the interface.

Volume tracking methods in the past have almost universally implemented Laplace's formula, $p_\ell = p_0 + \sigma \kappa$, as the interfacial boundary condition, neglecting the liquid phase deviatoric normal stress. In addition, currently available free surface computational techniques typically assume constant pressure in the vapor phase, a limitation that must be removed for accurate analysis of the liquid sheet instability problem where pressure variations in the vapor phase are the driving force for wave growth.

With these goals in mind, the principal objectives of this thesis are:

- Improve the accuracy of existing volume tracking techniques for free surface flows at Reynolds numbers where the accuracy of the advective terms is important.

- Implement the complete normal stress boundary condition at the free surface by extending Laplace's formula to include the deviatoric normal stress in the liquid phase at the interface.
- Add the capability to simultaneously solve the coupled problem of flow in a viscous liquid phase and an inviscid vapor phase.
- To demonstrate the applicability of this technique to additional problems of interest to the pulp and paper industry (e.g. three-dimensional flow under a coating blade and flow of condensate in a dryer cylinder).

NUMERICAL TECHNIQUE

In this section, I describe in detail the numerical schemes used in the IPST-VOF3D computational technique to solve the incompressible Navier-Stokes equations for flow problems having a free surface. I discuss the algorithm neglecting the interfacial deviatoric normal stress in the liquid phase and assuming the pressure in the vapor phase to be constant. In this form, the computational technique is essentially the NASA-VOF3D[81] computational technique. In the following section, I describe the major modifications necessary to include the effects of the deviatoric normal stress in the liquid phase on the interfacial boundary condition and to allow vapor phase pressure variations through solution of the potential flow equations in the vapor phase.

I begin by presenting the continuity equation for incompressible flow in a modified form. Since, in general, this computational technique can be used for either Cartesian or cylindrical coordinates, I will use the factors r and ζ to distinguish between the coordinate systems. The factors r = 1 and $\zeta = 0$ correspond to Cartesian coordinates, while r = x and $\zeta = 1$ correspond to cylindrical coordinates. Thus the continuity equation becomes

$$\nabla \cdot (\Theta \mathbf{u}) = \frac{1}{r}\frac{\partial (r\Theta u)}{\partial x} + \frac{1}{r}\frac{\partial (\Theta v)}{\partial y} + \frac{\partial (\Theta w)}{\partial z} = 0, \tag{31}$$

where u, v, and w are the velocity components in the x, y and z directions, respectively (or r, θ, and z which are represented as x, y, and z). The term Θ in Equation (31) arises from a partial cell treatment and is used to account for interior obstacles that block only a portion of a given computational cell.[80] The partial cell technique, discussed in greater detail below, allows more accurate simulation of interior obstacles with curved

boundaries than the "stair-step" approach often used in finite difference computational techniques.

Again using the r and ζ parameters, the momentum balance equations for incompressible Newtonian flow can be written as

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+\frac{v}{r}\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}-\zeta\frac{v^2}{r} = g_x-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left[\frac{\partial^2 u}{\partial x^2}+\frac{1}{r^2}\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+\zeta\left(\frac{1}{r}\frac{\partial u}{\partial x}-\frac{u}{r^2}-\frac{2}{r^2}\frac{\partial v}{\partial y}\right)\right], \tag{32}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+\frac{v}{r}\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}+\zeta\frac{u\,v}{r} = g_y-\frac{1}{\rho}\frac{1}{r}\frac{\partial p}{\partial y}+\nu\left[\frac{\partial^2 v}{\partial x^2}+\frac{1}{r^2}\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}+\zeta\left(\frac{1}{r}\frac{\partial v}{\partial x}-\frac{v}{r^2}+\frac{2}{r^2}\frac{\partial u}{\partial y}\right)\right], \tag{33}$$

and

$$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+\frac{v}{r}\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z} = g_z-\frac{1}{\rho}\frac{\partial p}{\partial z}+\nu\left[\frac{\partial^2 w}{\partial x^2}+\frac{1}{r^2}\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}+\zeta\frac{1}{r}\frac{\partial w}{\partial x}\right] \tag{34}$$

where g_x, g_y, and g_z are body forces in the x, y, and z directions, respectively.

The location of the liquid phase within the computational domain is tracked using the VOF technique as described above. The VOF "fullness" function, F, is transported through the domain by solution of the F-convection equation,

$$\frac{\partial(\Theta F)}{\partial t}=-\frac{1}{r}\frac{\partial(r\Theta F u)}{\partial x}-\frac{1}{r}\frac{\partial(\Theta F v)}{\partial y}-\frac{\partial(\Theta F w)}{\partial z}. \tag{35}$$

Now that I have presented the forms of the equations to be solved, I present the finite difference approximations of these equations. First, I describe the computational mesh used including a brief description of the mesh generation method (see Appendix I

documenting the code for more details). Next, I discuss the finite difference approximations leading to an explicit guess for the new velocity field forming the first step in a two-step projection method used to step forward in time, with special attention paid to the approximation of the advective terms in the NSE. This is followed by a description of the correction step, the second step of the projection and the two methods, Successive Over Relaxation (SOR) and the Conjugate Residual (CR) technique, available for solving the resulting Poisson pressure equation. Finally, I discuss the type of donor-acceptor differencing used to solve the F-convection equation which has remained unchanged from the NASA-VOF3D computational technique.

THE COMPUTATIONAL MESH

The computational mesh is a three-dimensional orthogonal mesh representing either Cartesian or cylindrical coordinates. For simplicity in the form of the equations, the cylindrical coordinate system is modified to use $x = r$, $y = r_{max}\theta$, and $z = z$ as the mapping between the coordinate systems with the terms in the continuity equation, the NSE, and the F-convection equation modified accordingly.

Solution is accomplished on a "staggered" grid where scalar quantities, such as the "fullness" function, F, and the pressure, p, are located at the computational cell centers and vector components such as the velocity components u, v, and w are located on the cell faces as shown in Figure 8 for a two-dimensional mesh. In the x-direction, the variable X(i) refers to the location of right side of the computational cell denoted $x_{i+\frac{1}{2}}$ in Figure 8. Similarly, XI(i) refers to the cell center, denoted x_i, DELX(i) refers to the local grid spacing, denoted $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$, and the distance between cell centers is denoted by $\Delta x_{i+\frac{1}{2}} = x_{i+1} - x_i$. The y and z directions are defined in an analogous manner.

Setup of the computational grid is accomplished through the input data. Each coordinate direction is divided into zones having a left boundary, XL; a right boundary, XR; a "focal" point, XC; the number of computational cells within the zone on either side of the focal point, NXL and NXR; and the spacing of the cells adjacent to the focal point, DXMN. If $DXMN > DX_{ave} = (XL - XC)/NXL$, the cells are equally spaced with a spacing of DX_{ave}. Otherwise, the cell spacing varies with the smallest cell adjacent to XC and the largest cells adjacent to XL and XR. The cell sizes are adjusted so that a smooth transition between cell sizes is maintained and $(DX_{XL} + DX_{XC})/2 = DX_{ave}$. Care must be taken to keep cells in adjacent zones of similar sizes and to maintain cell aspect ratios as close to unity as possible to maintain accuracy and stability.[81]

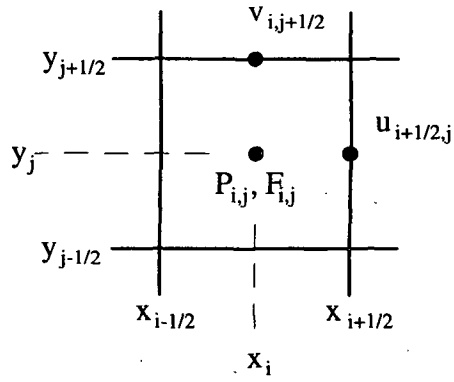

Figure 8. Diagram of a two-dimensional computational cell.

The remaining portion of the computational grid setup to be discussed is the partial cell treatment alluded to above. This computational technique uses a partial cell treatment for interior obstacles that eliminates the need for stair-stepping curved or diagonal boundaries as is often necessary in finite difference techniques. The basis of the partial cell treatment is discussed by Torrey et al.[80]

The partial areas open to flow, AR(i,j,k), ABK(i,j,k), and AT(i,j,k) representing the right, back and top boundaries, respectively, are stored at cell faces in the same manner as the velocity components. The cell volume open to flow, AC(i,j,k), is stored at the cell center in the manner of the scalar quantities F and P.

Values of AR, ABK, AT, and AC are set according to the interior obstacles defined in the input data. Assuming y or θ direction symmetry, a three-dimensional prism or conic section,

$$\Theta(x,z) \le a_1 x + a_2 x^2 + b_1 z + b_2 z^2 + c_1 + c_2 xz, \tag{36}$$

is defined. An input flag determines whether the volume contained within (36) is opened or closed to flow. From this function, the area of each cell boundary and the volume of the cell itself open or closed to flow may be calculated. By combining a series of obstacles, extremely complex geometries may be constructed, although currently they must be symmetric with respect to the y or θ direction. This procedure is demonstrated in the input data shown for subsequent test and example problems.

EXPLICIT PROJECTION STEP

The first step in the two-step projection consists of solving the equations resulting from explicit temporal finite differencing of the NSE. In vector form, the equation for the provisional velocity field is

$$\tilde{\mathbf{u}}^n = \mathbf{u}^n + \delta t\left[\mathbf{g} - \nabla P^n + \nu\nabla^2\mathbf{u}^n - \mathbf{u}^n \cdot \nabla\mathbf{u}^n\right]. \tag{37}$$

Note that from here on, the variable P refers to the reduced pressure, p/ρ.

Specifically, this step consists of adding a body acceleration, a specific pressure acceleration, an advection term, and a viscous acceleration computed using the previous velocity and pressure fields to the previous velocity field. These terms, computed in the

subroutine **TILDE** in the NASA-VOF3D program, will be explained below. In addition to the advective differencing scheme used in the NASA-VOF3D computational technique, three third order accurate finite difference schemes are presented below and available as options in the IPST-VOF3D program.

With the terms defined in the subsequent subsections, the x, y, and z components of the explicit projection step for the NSE become

$$\tilde{u}^n_{i+\frac{1}{2},j,k} = u^n_{i+\frac{1}{2},j,k} + \delta t\left[ACCX^n_{i+\frac{1}{2},j,k} - ADVX^n_{i+\frac{1}{2},j,k} + VISX^n_{i+\frac{1}{2},j,k}\right], \tag{38a}$$

$$\tilde{v}^n_{i,j+\frac{1}{2},k} = v^n_{i,j+\frac{1}{2},k} + \delta t\left[ACCY^n_{i,j+\frac{1}{2},k} - ADVY^n_{i,j+\frac{1}{2},k} + VISY^n_{i,j+\frac{1}{2},k}\right], \tag{38b}$$

and $$\tilde{w}^n_{i,j,k+\frac{1}{2}} = w^n_{i,j,k+\frac{1}{2}} + \delta t\left[ACCZ^n_{i,j,k+\frac{1}{2}} - ADVZ^n_{i,j,k+\frac{1}{2}} + VISZ^n_{i,j,k+\frac{1}{2}}\right] \tag{38c}$$

where the body forces and specific pressure accelerations have been lumped together in the terms ACCX, ACCY, and ACCZ.

Gravitational and Specific Pressure Accelerations

The body and specific pressure accelerations in the x, y, and z directions are

$$ACCX^n_{i+\frac{1}{2},j,k} = g_x + \left(P^n_{i+1,j,k} - P^n_{i,j,k}\right)\Big/\Delta x_{i+\frac{1}{2}}, \tag{39a}$$

$$ACCY^n_{i,j+\frac{1}{2},k} = g_y + \left(P^n_{i,j+1,k} - P^n_{i,j,k}\right)\Big/\left(r_i \Delta y_{j+\frac{1}{2}}\right), \tag{39b}$$

and $$ACCZ^n_{i,j,k+\frac{1}{2}} = g_z + \left(P^n_{i,j,k+1} - P^n_{i,j,k}\right)\Big/\Delta z_{k+\frac{1}{2}}, \tag{39c}$$

respectively. As above, the term r_i is 1 for Cartesian coordinates and x_i for cylindrical coordinates. In cylindrical coordinates the body accelerations in the r and θ direction are typically not constant, but a function of position.

Viscous Acceleration

The viscous accelerations in the x-direction is computed as a second partial derivative. Therefore, several first partial derivatives near the computational cell of interest need to be computed. These first partial derivatives for the u-component of velocity, used here and below, are defined as:

$$\left(\frac{\partial u}{\partial x}\right)^n_{i,j,k} = \left(u^n_{i+\frac{1}{2},j,k} - u^n_{i-\frac{1}{2},j,k}\right)\Big/\Delta x_i\,, \tag{40a}$$

$$\left(\frac{\partial u}{\partial x}\right)^n_{i+1,j,k} = \left(u^n_{i+\frac{3}{2},j,k} - u^n_{i+\frac{1}{2},j,k}\right)\Big/\Delta x_{i+1}\,, \tag{40b}$$

$$\left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} = \left(u^n_{i+\frac{1}{2},j+1,k} - u^n_{i+\frac{1}{2},j,k}\right)\Big/\left(r_{i+\frac{1}{2}}\Delta y_{j+\frac{1}{2}}\right), \tag{40c}$$

$$\left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j-\frac{1}{2},k} = \left(u^n_{i+\frac{1}{2},j,k} - u^n_{i+\frac{1}{2},j-1,k}\right)\Big/\left(r_{i+\frac{1}{2}}\Delta y_{j-\frac{1}{2}}\right), \tag{40d}$$

$$\left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k+\frac{1}{2}} = \left(u^n_{i+\frac{1}{2},j,k+1} - u^n_{i+\frac{1}{2},j,k}\right)\Big/\Delta z_{k+\frac{1}{2}}\,, \tag{40e}$$

and $$\left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k-\frac{1}{2}} = \left(u^n_{i+\frac{1}{2},j,k} - u^n_{i+\frac{1}{2},j,k-1}\right)\Big/\Delta z_{k-\frac{1}{2}}\,. \tag{40f}$$

Similarly the finite differences in the v velocity component are

$$\left(\frac{\partial v}{\partial x}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} = \left(v^n_{i+1,j+\frac{1}{2},k} - v^n_{i,j+\frac{1}{2},k}\right)\Big/\Delta x_{i+\frac{1}{2}}\,, \tag{41a}$$

$$\left(\frac{\partial v}{\partial x}\right)^n_{i-\frac{1}{2},j+\frac{1}{2},k} = \left(v^n_{i,j+\frac{1}{2},k} - v^n_{i-1,j+\frac{1}{2},k}\right)\Big/\Delta x_{i-\frac{1}{2}}\,, \tag{41b}$$

$$\left(\frac{\partial v}{\partial y}\right)^n_{i,j,k} = \left(v^n_{i,j+\frac{1}{2},k} - v^n_{i,j-\frac{1}{2},k}\right)\Big/\left(r_i\Delta y_j\right), \tag{41c}$$

$$\left(\frac{\partial v}{\partial y}\right)^n_{i,j+1,k} = \left(v^n_{i,j+\frac{3}{2},k} - v^n_{i,j+\frac{1}{2},k}\right)\Big/\left(r_i\Delta y_{j+1}\right), \tag{41d}$$

$$\left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k+\frac{1}{2}} = \left(v^n_{i,j+\frac{1}{2},k+1} - v^n_{i,j+\frac{1}{2},k}\right)\Big/\Delta z_{k+\frac{1}{2}}\,, \tag{41e}$$

and $$\left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k-\frac{1}{2}} = \left(v^n_{i,j+\frac{1}{2},k} - v^n_{i,j+\frac{1}{2},k-1}\right)\Big/\Delta z_{k-\frac{1}{2}}\,. \tag{41f}$$

Finally, the finite differences for the w velocity component are

$$\left(\frac{\partial w}{\partial x}\right)^n_{i+\frac{1}{2},j,k+\frac{1}{2}} = \left(w^n_{i+1,j,k+\frac{1}{2}} - w^n_{i,j,k+\frac{1}{2}}\right)\Big/\Delta x_{i+\frac{1}{2}}, \tag{42a}$$

$$\left(\frac{\partial w}{\partial x}\right)^n_{i-\frac{1}{2},j,k+\frac{1}{2}} = \left(v^n_{i,j,k+\frac{1}{2}} - v^n_{i-1,j,k+\frac{1}{2}}\right)\Big/\Delta x_{i-\frac{1}{2}}, \tag{42b}$$

$$\left(\frac{\partial w}{\partial y}\right)^n_{i,j+\frac{1}{2},k+\frac{1}{2}} = \left(w^n_{i,j+1,k+\frac{1}{2}} - w^n_{i,j,k+\frac{1}{2}}\right)\Big/\left(r_i\Delta y_{j+\frac{1}{2}}\right), \tag{42c}$$

$$\left(\frac{\partial w}{\partial y}\right)^n_{i,j-\frac{1}{2},k+\frac{1}{2}} = \left(w^n_{i,j,k+\frac{1}{2}} - w^n_{i,j-1,k+\frac{1}{2}}\right)\Big/\left(r_i\Delta y_{j-\frac{1}{2}}\right), \tag{42d}$$

$$\left(\frac{\partial w}{\partial z}\right)^n_{i,j,k} = \left(w^n_{i,j,k+\frac{1}{2}} - w^n_{i,j,k-\frac{1}{2}}\right)\Big/\Delta z_k, \tag{42e}$$

and

$$\left(\frac{\partial w}{\partial z}\right)^n_{i,j,k+1} = \left(w^n_{i,j,k+\frac{3}{2}} - w^n_{i,j,k+\frac{1}{2}}\right)\Big/\Delta z_{k+1}. \tag{42f}$$

From these definitions, the viscous accelerations are computed using second order accurate finite difference formulas,

$$\begin{aligned} VISX^n_{i+\frac{1}{2},j,k} = \nu\Bigg\{ & \frac{\left(\frac{\partial u}{\partial x}\right)^n_{i+1,j,k} - \left(\frac{\partial u}{\partial x}\right)^n_{i,j,k}}{\Delta x_{i+\frac{1}{2}}} + \frac{\left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} - \left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j-\frac{1}{2},k}}{\Delta y_j} + \frac{\left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k+\frac{1}{2}} - \left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k-\frac{1}{2}}}{\Delta z_k} \\ & + \zeta\left[\frac{\Delta x_i\left(\frac{\partial u}{\partial x}\right)^n_{i+1,j,k} + \Delta x_{i+1}\left(\frac{\partial u}{\partial x}\right)^n_{i,j,k}}{x_{i+\frac{1}{2}}\left(\Delta x_i + \Delta x_{i+1}\right)} - \frac{u^n_{i+\frac{1}{2},j,k}}{x^2_{i+\frac{1}{2}}} - 2\frac{\Delta x_i\left(\frac{\partial v}{\partial y}\right)^n_{i+1,j,k} + \Delta x_{i+1}\left(\frac{\partial v}{\partial y}\right)^n_{i,j,k}}{x^2_{i+\frac{1}{2}}\left(\Delta x_i + \Delta x_{i+1}\right)}\right]\Bigg\}, \end{aligned} \tag{43a}$$

$$\begin{aligned} VISY^n_{i,j+\frac{1}{2},k} = \nu\Bigg\{ & \frac{\left(\frac{\partial v}{\partial x}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} - \left(\frac{\partial v}{\partial x}\right)^n_{i-\frac{1}{2},j+\frac{1}{2},k}}{\Delta x_i} + \frac{\left(\frac{\partial v}{\partial y}\right)^n_{i,j+1,k} - \left(\frac{\partial v}{\partial y}\right)^n_{i,j,k}}{\Delta y_{j+\frac{1}{2}}} + \frac{\left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k+\frac{1}{2}} - \left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k-\frac{1}{2}}}{\Delta z_k} \\ & + \zeta\left[\frac{\left(\frac{\partial v}{\partial x}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} + \left(\frac{\partial v}{\partial x}\right)^n_{i-\frac{1}{2},j+\frac{1}{2},k}}{2x_i} - \frac{v^n_{i,j+\frac{1}{2},k}}{x^2_i} + \frac{u^n_{i+\frac{1}{2},j+1,k} + u^n_{i-\frac{1}{2},j+1,k} - u^n_{i+\frac{1}{2},j,k} - u^n_{i-\frac{1}{2},j,k}}{x^2_i\Delta y_{i+\frac{1}{2}}}\right]\Bigg\}, \end{aligned} \tag{43b}$$

and

$$VISZ^{n}_{i,j,k+\frac{1}{2}} = \nu\left\{\frac{\left(\frac{\partial w}{\partial x}\right)^{n}_{i+\frac{1}{2},j,k+\frac{1}{2}} - \left(\frac{\partial w}{\partial x}\right)^{n}_{i-\frac{1}{2},j,k+\frac{1}{2}}}{\Delta x_i} + \frac{\left(\frac{\partial w}{\partial y}\right)^{n}_{i,j+\frac{1}{2},k+\frac{1}{2}} - \left(\frac{\partial w}{\partial y}\right)^{n}_{i,j-\frac{1}{2},k+\frac{1}{2}}}{\Delta y_j} + \frac{\left(\frac{\partial w}{\partial z}\right)^{n}_{i,j,k+1} - \left(\frac{\partial w}{\partial z}\right)^{n}_{i,j,k}}{\Delta z_{k+\frac{1}{2}}}\right.$$
$$\left. + \zeta\frac{\left(\frac{\partial w}{\partial x}\right)^{n}_{i+\frac{1}{2},j,k+\frac{1}{2}} + \left(\frac{\partial w}{\partial x}\right)^{n}_{i-\frac{1}{2},j,k+\frac{1}{2}}}{2x_i}\right\}. \qquad (43c)$$

Advective Terms

For numerical stability, the advective terms in the NSE must be treated using finite difference formulas with some level of upwind differencing, where the advective differences are weighted toward the local upstream side in some manner. Using central differences that are equally weighted between the upstream and downstream values leads to an unconditionally unstable numerical scheme and must be avoided.[84] The original implementations based on the SOLA formulation used a linear combination of first order accurate upwind differencing and second order accurate central differencing (from here on this combination is termed SOLA differencing) to achieve a combination of the numerical stability of the upwind scheme and the accuracy of the second order scheme. Since the previous terms in the NSE, with the exception of the temporal differences, use second order accurate finite difference schemes, the accuracy of a simulation at increasing Reynolds number is limited by the formally first order accurate treatment of the advective terms. In order to improve the accuracy of simulations at high Reynolds number while maintaining numerical stability, I have implemented three different third order accurate finite difference schemes for the advective terms; quadratic upwind differencing for convective kinematics (QUICK),[85] third order accurate upwind differencing (THIRD),[86] and the method of Kawamura and Kuwahara (KANDK).[87]

Here, I describe the finite difference representations for the advective term differencing options. The details of the derivations of the third order accurate schemes can be found in Appendix II, Appendix III, and Appendix IV for QUICK, THIRD, and KANDK, respectively. First, I present the finite difference formulas for the advective terms in the NSE as implemented using SOLA differencing. This is followed by descriptions of the terms that are different when each of the third order accurate finite difference schemes are used.

SOLA Differencing:

The x-direction advective term can be written as

$$ADVX^n_{i+\frac{1}{2},j,k} = \left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} + \frac{1}{r_{i+\frac{1}{2}}}\left(v\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} + \left(w\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k} - \zeta\frac{v^{n\,2}_{i+\frac{1}{2},j,k}}{x_{i+\frac{1}{2}}}, \tag{44}$$

where
$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = u^n_{i+\frac{1}{2},j,k}\left[\frac{(1-\alpha')\Delta x_i\left(\frac{\partial u}{\partial x}\right)^n_{i+1,j,k} + (1+\alpha')\Delta x_{i+1}\left(\frac{\partial u}{\partial x}\right)^n_{i,j,k}}{(1-\alpha')\Delta x_i + (1+\alpha')\Delta x_{i+1}}\right], \tag{45a}$$

$$\left(v\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} = v^n_{i+\frac{1}{2},j,k}\left[\frac{(1-\alpha')\Delta y_{j-\frac{1}{2}}\left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} + (1+\alpha')\Delta y_{j+\frac{1}{2}}\left(\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j-\frac{1}{2},k}}{(1-\alpha')\Delta y_{j-\frac{1}{2}} + (1+\alpha')\Delta y_{j+\frac{1}{2}}}\right], \tag{45b}$$

$$\left(w\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} = w^n_{i+\frac{1}{2},j,k}\left[\frac{(1-\alpha')\Delta z_{k-\frac{1}{2}}\left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k+\frac{1}{2}} + (1+\alpha')\Delta z_{k+\frac{1}{2}}\left(\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k-\frac{1}{2}}}{(1-\alpha')\Delta z_{k-\frac{1}{2}} + (1+\alpha')\Delta z_{k+\frac{1}{2}}}\right], \tag{45c}$$

$$v^n_{i+\frac{1}{2},j,k} = \frac{\Delta x_i\left(v^n_{i+1,j+\frac{1}{2},k} + v^n_{i+1,j-\frac{1}{2},k}\right) + \Delta x_{i+1}\left(v^n_{i,j+\frac{1}{2},k} + v^n_{i,j-\frac{1}{2},k}\right)}{2\left(\Delta x_i + \Delta x_{i+1}\right)}, \tag{45d}$$

and
$$w^n_{i+\frac{1}{2},j,k} = \frac{\Delta x_i\left(w^n_{i+1,j,k+\frac{1}{2}} + w^n_{i+1,j,k-\frac{1}{2}}\right) + \Delta x_{i+1}\left(w^n_{i,j,k+\frac{1}{2}} + w^n_{i,j,k-\frac{1}{2}}\right)}{2\left(\Delta x_i + \Delta x_{i+1}\right)}. \tag{45e}$$

As an example, I use the term

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = \frac{u^n_{i+\frac{1}{2},j,k}}{2}\left[(1-\alpha')(\partial u/\partial x)^n_{i+1,j,k} + (1+\alpha')(\partial u/\partial x)^n_{i,j,k}\right], \tag{46}$$

here rewritten for a constant grid, to describe SOLA differencing. The parameter α' is derived from the fraction of upwind differencing, α,

$$\alpha' = \alpha\,\mathrm{sign}\left(u^n_{i+\frac{1}{2},j,k}\right). \tag{47}$$

Thus, if $u^n_{i+\frac{1}{2},j,k} > 0$ and $\alpha = 1$, then (46) becomes

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = u^n_{i+\frac{1}{2},j,k}\left(\frac{\partial u}{\partial x}\right)^n_{i,j,k}, \tag{48}$$

which is pure first order accurate upwind differencing. Similarly, if $u^n_{i+\frac{1}{2},j,k} > 0$ and $\alpha = 0$, then (46) becomes

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = \tfrac{1}{2}u^n_{i+\frac{1}{2},j,k}\left[\left(\frac{\partial u}{\partial x}\right)^n_{i+1,j,k} + \left(\frac{\partial u}{\partial x}\right)^n_{i,j,k}\right], \tag{49}$$

which is second order accurate central differencing (an unstable numerical scheme).

Next, the y-direction advective term can be written as

$$\mathrm{ADVY}^n_{i,j+\frac{1}{2},k} = \left(u\frac{\partial v}{\partial x}\right)^n_{i,j+\frac{1}{2},k} + \frac{1}{r_i}\left(v\frac{\partial v}{\partial y}\right)^n_{i,j+\frac{1}{2},k} + \left(w\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k} - \zeta\frac{u^n_{i,j+\frac{1}{2},k}v^n_{i,j+\frac{1}{2},k}}{x_i}, \tag{50}$$

where

$$\left(u\frac{\partial v}{\partial x}\right)^n_{i,j+\frac{1}{2},k} = u^n_{i,j+\frac{1}{2},k}\left[\frac{(1-\alpha')\Delta x_{i-\frac{1}{2}}\left(\frac{\partial v}{\partial x}\right)^n_{i+\frac{1}{2},j+\frac{1}{2},k} + (1+\alpha')\Delta x_{i+\frac{1}{2}}\left(\frac{\partial v}{\partial x}\right)^n_{i-\frac{1}{2},j+\frac{1}{2},k}}{(1-\alpha')\Delta x_{i-\frac{1}{2}} + (1+\alpha')\Delta x_{i+\frac{1}{2}}}\right], \tag{51a}$$

$$\left(v\frac{\partial v}{\partial y}\right)^n_{i,j+\frac{1}{2},k} = v^n_{i,j+\frac{1}{2},k}\left[\frac{(1-\alpha')\Delta y_j\left(\frac{\partial v}{\partial y}\right)^n_{i,j+1,k} + (1+\alpha')\Delta y_{j+1}\left(\frac{\partial v}{\partial y}\right)^n_{i,j,k}}{(1-\alpha')\Delta y_j + (1+\alpha')\Delta y_{j+1}}\right], \tag{51b}$$

$$\left(w\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k} = w^n_{i,j+\frac{1}{2},k}\left[\frac{(1-\alpha')\Delta z_{i-\frac{1}{2}}\left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k+\frac{1}{2}} + (1+\alpha')\Delta z_{i+\frac{1}{2}}\left(\frac{\partial v}{\partial z}\right)^n_{i,j+\frac{1}{2},k-\frac{1}{2}}}{(1-\alpha')\Delta z_{i-\frac{1}{2}} + (1+\alpha')\Delta z_{i+\frac{1}{2}}}\right], \tag{51c}$$

$$u^n_{i,j+\frac{1}{2},k} = \frac{\Delta y_j\left(u^n_{i+\frac{1}{2},j+1,k} + u^n_{i-\frac{1}{2},j+1,k}\right) + \Delta y_{j+1}\left(u^n_{i+\frac{1}{2},j,k} + u^n_{i-\frac{1}{2},j,k}\right)}{2\left(\Delta y_j + \Delta y_{j+1}\right)}, \tag{51d}$$

and

$$w^n_{i,j+\frac{1}{2},k} = \frac{\Delta y_j\left(w^n_{i,j+1,k+\frac{1}{2}} + w^n_{i,j+1,k-\frac{1}{2}}\right) + \Delta y_{j+1}\left(w^n_{i,j,k+\frac{1}{2}} + w^n_{i,j,k-\frac{1}{2}}\right)}{2\left(\Delta y_j + \Delta y_{j+1}\right)}. \tag{51e}$$

Finally, the z-direction advective term can be written as

$$ADVZ^n_{i,j,k+\frac{1}{2}} = \left(u\frac{\partial w}{\partial x}\right)^n_{i,j,k+\frac{1}{2}} + \frac{1}{r_i}\left(v\frac{\partial w}{\partial y}\right)^n_{i,j,k+\frac{1}{2}} + \left(w\frac{\partial w}{\partial z}\right)^n_{i,j,k+\frac{1}{2}}, \tag{52}$$

where

$$\left(u\frac{\partial w}{\partial x}\right)^n_{i,j,k+\frac{1}{2}} = u^n_{i,j,k+\frac{1}{2}}\left[\frac{(1-\alpha')\Delta x_{i-\frac{1}{2}}\left(\frac{\partial w}{\partial x}\right)^n_{i+\frac{1}{2},j,k+\frac{1}{2}} + (1+\alpha')\Delta x_{i+\frac{1}{2}}\left(\frac{\partial w}{\partial x}\right)^n_{i-\frac{1}{2},j,k+\frac{1}{2}}}{(1-\alpha')\Delta x_{i-\frac{1}{2}} + (1+\alpha')\Delta x_{i+\frac{1}{2}}}\right], \tag{53a}$$

$$\left(v\frac{\partial w}{\partial y}\right)^n_{i,j,k+\frac{1}{2}} = v^n_{i,j,k+\frac{1}{2}}\left[\frac{(1-\alpha')\Delta y_{j-\frac{1}{2}}\left(\frac{\partial w}{\partial y}\right)^n_{i,j+\frac{1}{2},k+\frac{1}{2}} + (1+\alpha')\Delta y_{j+\frac{1}{2}}\left(\frac{\partial w}{\partial y}\right)^n_{i,j-\frac{1}{2},k+\frac{1}{2}}}{(1-\alpha')\Delta y_{j-\frac{1}{2}} + (1+\alpha')\Delta y_{j+\frac{1}{2}}}\right], \tag{53b}$$

$$\left(w\frac{\partial w}{\partial z}\right)^n_{i,j,k+\frac{1}{2}} = w^n_{i,j,k+\frac{1}{2}}\left[\frac{(1-\alpha')\Delta z_k\left(\frac{\partial w}{\partial z}\right)^n_{i,j,k+1} + (1+\alpha')\Delta z_{k+1}\left(\frac{\partial w}{\partial z}\right)^n_{i,j,k}}{(1-\alpha')\Delta z_k + (1+\alpha')\Delta z_{k+1}}\right], \tag{53c}$$

$$u^n_{i,j,k+\frac{1}{2}} = \frac{\Delta z_k\left(u^n_{i+\frac{1}{2},j,k+1} + u^n_{i-\frac{1}{2},j,k+1}\right) + \Delta z_{k+1}\left(u^n_{i+\frac{1}{2},j,k} + u^n_{i-\frac{1}{2},j,k}\right)}{2\left(\Delta z_k + \Delta z_{k+1}\right)}, \tag{53d}$$

and

$$v^n_{i,j,k+\frac{1}{2}} = \frac{\Delta z_k\left(v^n_{i,j+\frac{1}{2},k+1} + v^n_{i,j-\frac{1}{2},k+1}\right) + \Delta z_{k+1}\left(v^n_{i,j+\frac{1}{2},k} + v^n_{i,j-\frac{1}{2},k}\right)}{2\left(\Delta z_k + \Delta z_{k+1}\right)}. \tag{53e}$$

QUICK Differencing:

For QUICK differencing, the situation is a bit more complex. The finite difference formulas for variable grids are derived in Appendix II; however, for simplicity, only the constant grid formulas are presented here. The variable grid formula uses the same velocities with weighting factors that depend on the local grid spacings.

Many of the components of the advective terms remain the same as in the SOLA differencing just presented and only the terms that vary are presented. With constant grid spacings represented as δx, δy, and δz, the resulting formulas are

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{u^n_{i+\frac{1}{2},j,k}}{8\delta x}\left(3u^n_{i+\frac{3}{2},j,k} + 3u^n_{i+\frac{1}{2},j,k} - 7u^n_{i-\frac{1}{2},j,k} + u^n_{i-\frac{3}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{u^n_{i+\frac{1}{2},j,k}}{8\delta x}\left(-u^n_{i+\frac{5}{2},j,k} + 7u^n_{i+\frac{3}{2},j,k} - 3u^n_{i+\frac{1}{2},j,k} - 3u^n_{i-\frac{1}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k} < 0 \end{cases}, \tag{54a}$$

$$\left(v\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{v^n_{i+\frac{1}{2},j,k}}{8\delta y}\left(3u^n_{i+\frac{1}{2},j+1,k} + 3u^n_{i+\frac{1}{2},j,k} - 7u^n_{i+\frac{1}{2},j-1,k} + u^n_{i+\frac{1}{2},j-2,k}\right) & v^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{v^n_{i+\frac{1}{2},j,k}}{8\delta y}\left(-u^n_{i+\frac{1}{2},j+2,k} + 7u^n_{i+\frac{1}{2},j+1,k} - 3u^n_{i+\frac{1}{2},j,k} - 3u^n_{i+\frac{1}{2},j-1,k}\right) & v^n_{i+\frac{1}{2},j,k} < 0 \end{cases}, \tag{54b}$$

and

$$\left(w\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{w^n_{i+\frac{1}{2},j,k}}{8\delta z}\left(3u^n_{i+\frac{1}{2},j,k+1} + 3u^n_{i+\frac{1}{2},j,k} - 7u^n_{i+\frac{1}{2},j,k-1} + u^n_{i+\frac{1}{2},j,k-2}\right) & w^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{w^n_{i+\frac{1}{2},j,k}}{8\delta z}\left(-u^n_{i+\frac{1}{2},j,k+2} + 7u^n_{i+\frac{1}{2},j,k+1} - 3u^n_{i+\frac{1}{2},j,k} - 3u^n_{i+\frac{1}{2},j,k-1}\right) & w^n_{i+\frac{1}{2},j,k} < 0 \end{cases} \tag{54c}$$

with analogous formulas for the equations in the y and z-directions.

Third Order Accurate Upwind Differencing

The complete formulas for third order accurate upwind differencing with variable grid spacing can be found in Appendix III. As with the quick differencing above, the formulas presented here are for constant grid spacings only:

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{u^n_{i+\frac{1}{2},j,k}}{6\delta x}\left(2u^n_{i+\frac{3}{2},j,k}+3u^n_{i+\frac{1}{2},j,k}-6u^n_{i-\frac{1}{2},j,k}+u^n_{i-\frac{3}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k}>0 \\ \dfrac{u^n_{i+\frac{1}{2},j,k}}{6\delta x}\left(-u^n_{i+\frac{5}{2},j,k}+6u^n_{i+\frac{3}{2},j,k}-3u^n_{i+\frac{1}{2},j,k}-2u^n_{i-\frac{1}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k}<0 \end{cases} \quad (55a)$$

$$\left(v\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{v^n_{i+\frac{1}{2},j,k}}{6\delta y}\left(2u^n_{i+\frac{1}{2},j+1,k}+3u^n_{i+\frac{1}{2},j,k}-6u^n_{i+\frac{1}{2},j-1,k}+u^n_{i+\frac{1}{2},j-2,k}\right) & v^n_{i+\frac{1}{2},j,k}>0 \\ \dfrac{v^n_{i+\frac{1}{2},j,k}}{6\delta y}\left(-u^n_{i+\frac{1}{2},j+2,k}+6u^n_{i+\frac{1}{2},j+1,k}-3u^n_{i+\frac{1}{2},j,k}-2u^n_{i+\frac{1}{2},j-1,k}\right) & v^n_{i+\frac{1}{2},j,k}<0 \end{cases} \quad (55b)$$

and

$$\left(w\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{w^n_{i+\frac{1}{2},j,k}}{6\delta z}\left(2u^n_{i+\frac{1}{2},j,k+1}+3u^n_{i+\frac{1}{2},j,k}-6u^n_{i+\frac{1}{2},j,k-1}+u^n_{i+\frac{1}{2},j,k-2}\right) & w^n_{i+\frac{1}{2},j,k}>0 \\ \dfrac{w^n_{i+\frac{1}{2},j,k}}{6\delta z}\left(-u^n_{i+\frac{1}{2},j,k+2}+6u^n_{i+\frac{1}{2},j,k+1}-3u^n_{i+\frac{1}{2},j,k}-2u^n_{i+\frac{1}{2},j,k-1}\right) & w^n_{i+\frac{1}{2},j,k}<0 \end{cases} \quad (55c)$$

with analogous formulas for the equations in the y and z-directions.

The Method of Kawamura and Kuwahara

Kawamura and Kuwahara's technique presents an additional challenge due to the nature of their derivation. For a constant grid they obtained the finite difference formula presented below. Unfortunately, I was unable to reproduce their derivation exactly on a variable grid. An approximate derivation of a variable grid version of Kawamura and Kuwahara's technique is presented in Appendix IV which is the numerical technique

present in the IPST-VOF3D program. For a constant grid, the formulas derived in Appendix IV reduce to these formulas of Kawamura and Kuwahara:

$$\left(u\frac{\partial u}{\partial x}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{u^n_{i+\frac{1}{2},j,k}}{6\delta x}\left(u^n_{i+\frac{5}{2},j,k} - 2u^n_{i+\frac{3}{2},j,k} + 9u^n_{i+\frac{1}{2},j,k} - 10u^n_{i-\frac{1}{2},j,k} + 2u^n_{i-\frac{3}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{u^n_{i+\frac{1}{2},j,k}}{6\delta x}\left(-2u^n_{i+\frac{5}{2},j,k} + 10u^n_{i+\frac{3}{2},j,k} - 9u^n_{i+\frac{1}{2},j,k} + 2u^n_{i-\frac{1}{2},j,k} - u^n_{i-\frac{3}{2},j,k}\right) & u^n_{i+\frac{1}{2},j,k} < 0 \end{cases}, \tag{56a}$$

$$\left(v\frac{\partial u}{\partial y}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{v^n_{i+\frac{1}{2},j,k}}{6\delta y}\left(u^n_{i+\frac{1}{2},j+2,k} - 2u^n_{i+\frac{1}{2},j+1,k} + 9u^n_{i+\frac{1}{2},j,k} - 10u^n_{i+\frac{1}{2},j-1,k} + 2u^n_{i+\frac{1}{2},j-2,k}\right) & v^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{v^n_{i+\frac{1}{2},j,k}}{6\delta y}\left(-2u^n_{i+\frac{1}{2},j+2,k} + 10u^n_{i+\frac{1}{2},j+1,k} - 9u^n_{i+\frac{1}{2},j,k} + 2u^n_{i+\frac{1}{2},j-1,k} - u^n_{i+\frac{1}{2},j-2,k}\right) & v^n_{i+\frac{1}{2},j,k} < 0 \end{cases}, \tag{56b}$$

and

$$\left(w\frac{\partial u}{\partial z}\right)^n_{i+\frac{1}{2},j,k} = \begin{cases} \dfrac{w^n_{i+\frac{1}{2},j,k}}{6\delta z}\left(u^n_{i+\frac{1}{2},j,k+2} - 2u^n_{i+\frac{1}{2},j,k+1} + 9u^n_{i+\frac{1}{2},j,k} - 10u^n_{i+\frac{1}{2},j,k-1} + 2u^n_{i+\frac{1}{2},j,k-2}\right) & w^n_{i+\frac{1}{2},j,k} > 0 \\ \dfrac{w^n_{i+\frac{1}{2},j,k}}{6\delta z}\left(-2u^n_{i+\frac{1}{2},j,k+2} + 10u^n_{i+\frac{1}{2},j,k+1} - 9u^n_{i+\frac{1}{2},j,k} + 2u^n_{i+\frac{1}{2},j,k-1} - u^n_{i+\frac{1}{2},j,k-2}\right) & w^n_{i+\frac{1}{2},j,k} < 0 \end{cases} \tag{56c}$$

with analogous formulas for the equations in the y and z-directions.

THE PRESSURE CORRECTION STEP

The correction step in the two-step projection method accomplishes an implicit correction of the pressure field based on the explicit guess for the velocity field computed in the first projection step. The velocity after a time step is defined as the provisional velocity computed in the first projection step plus a correction for the pressure change across the time step,

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^n - \delta t\, \nabla\left(\delta P^{n+1}\right). \tag{57}$$

Since the continuity equation must be satisfied at all times, the new velocity field can be substituted into the continuity equation,

$$\nabla\cdot\left(\Theta\mathbf{u}^{n+1}\right) = 0, \tag{58}$$

resulting in a Poisson equation for pressure (PPE),

$$\delta t \; V \nabla \cdot \left[\Theta \nabla \left(\delta P^{n+1}\right)\right] = V \nabla \cdot \left(\Theta \tilde{u}^{n}\right), \tag{59}$$

where V is the volume of the computational cell needed to ensure a symmetric system of equations.[80] Below, two numerical schemes for the solution of the PPE are presented, but first the pressure boundary condition adjacent to a no-slip wall must be derived.

The boundary condition for the pressure on a no-slip wall at the left edge (smallest value of x) of the computational cell (2,j) with constant cell spacing, is derived from Equation 57

$$\frac{\delta P_{2,j}^{n+1} - \delta P_{1,j}^{n+1}}{\Delta x} = -\frac{1}{\delta t}\left(u_{\frac{3}{2},j}^{n+1} - \tilde{u}_{\frac{3}{2},j}\right). \tag{60}$$

For a computational cell adjacent to the boundary, the two-dimensional finite difference form of the PPE (assuming constant cell volume and all cells open to flow) is

$$\begin{aligned} &\frac{1}{\Delta x}\left(\frac{\delta P_{3,j}^{n+1} - \delta P_{2,j}^{n+1}}{\Delta x} - \frac{\delta P_{2,j}^{n+1} - \delta P_{1,j}^{n+1}}{\Delta x}\right) + \frac{1}{\Delta y}\left(\frac{\delta P_{2,j+1}^{n+1} - \delta P_{2,j}^{n+1}}{\Delta y} - \frac{\delta P_{2,j}^{n+1} - \delta P_{2,j-1}^{n+1}}{\Delta y}\right) \\ &\qquad = \frac{1}{\Delta t}\left(\frac{\tilde{u}_{\frac{3}{2},j} - \tilde{u}_{x=x_L,j}}{\Delta x} + \frac{\tilde{v}_{2,j+\frac{1}{2}} - \tilde{v}_{2,j-\frac{1}{2}}}{\Delta y}\right). \end{aligned} \tag{61}$$

Upon substitution of Equation 60 into Equation 61 it is apparent that the solution of the PPE is independent of the value of $\tilde{u}_{x=x_L,j} = \tilde{u}_{\frac{3}{2},j}$ so it can be set equal to $u_{\frac{3}{2},j}^{n+1}$ yielding the boundary condition $\partial(\delta P)/\partial x = 0$.[88, 89] This condition can be applied to any domain boundary since the only requirement is that solution of the PPE is independent of the provisional velocity at the boundary. However, as Peyret and Taylor[88] point out, "it must be clear that this zero-derivative condition is purely numerical and does not imply that the real pressure gradient is zero."

SOR Option for the Second Projection Step

The SOR option does not solve the PPE directly, but accomplishes the same purpose. The conservation of mass at any time step is computed from Equation (31). In finite difference form this is represented as

$$D_{i,j,k} = \frac{1}{AC_{i,j,k}}\left[\frac{r_{i+\frac{1}{2}}AR_{i+\frac{1}{2},j,k}u_{i+\frac{1}{2},j,k} - r_{i-\frac{1}{2}}AR_{i-\frac{1}{2},j,k}u_{i-\frac{1}{2},j,k}}{r_i\Delta x_i} + \frac{ABK_{i,j+\frac{1}{2},k}v_{i,j+\frac{1}{2},k} - ABK_{i,j-\frac{1}{2},k}v_{i,j-\frac{1}{2},k}}{r_i\Delta y_j} + \frac{AT_{i,j,k+\frac{1}{2}}w_{i,j,k+\frac{1}{2}} - AT_{i,j,k-\frac{1}{2}}w_{i,j,k-\frac{1}{2}}}{\Delta z_k}\right] \quad (62)$$

where the divergence, $D_{i,j,k}$, is the error in satisfying the continuity equation (ideally this should be zero for incompressible flow).

With this definition in mind, an iteration procedure for updating the pressure and the velocity components is described. First, the pressure after the iteration is defined as

$$P^{v}_{i,j,k} = P^{v-1}_{i,j,k} + \delta P^{v}_{i,j,k} \quad (63)$$

where $\delta P^{v}_{i,j,k} = -\beta D^{v}_{i,j,k}$, (64a)

and

$$\frac{1}{\beta} = \frac{\delta t}{AC_{i,j,k}}\left\{\frac{1}{\Delta x_i}\left[\frac{AR_{i+\frac{1}{2},j,k}}{\Delta x_{i+\frac{1}{2}}} + \frac{AR_{i-\frac{1}{2},j,k}}{\Delta x_{i-\frac{1}{2}}}\right] + \frac{\zeta}{2x_i}\left[\frac{AR_{i+\frac{1}{2},j,k}}{\Delta x_{i+\frac{1}{2}}} - \frac{AR_{i-\frac{1}{2},j,k}}{\Delta x_{i-\frac{1}{2}}}\right] + \frac{1}{r_i\Delta y_j}\left[\frac{ABK_{i,j+\frac{1}{2},k}}{\Delta y_{j+\frac{1}{2}}} + \frac{ABK_{i,j-\frac{1}{2},k}}{\Delta y_{j-\frac{1}{2}}}\right] + \frac{1}{\Delta z_k}\left[\frac{AT_{i,j,k+\frac{1}{2}}}{\Delta z_{k+\frac{1}{2}}} + \frac{AT_{i,j-\frac{1}{2},k}}{\Delta z_{k-\frac{1}{2}}}\right]\right\}. \quad (64b)$$

Once the new value for the pressure has been determined from Equation (63), the velocities are updated:

$$u^{v}_{i+\frac{1}{2},j,k} = u^{v-1}_{i+\frac{1}{2},j,k} + \delta t\,\delta P^{v}_{i,j,k}/\Delta x_{i+\frac{1}{2}}, \quad (65a)$$

$$u^{v}_{i-\frac{1}{2},j,k} = u^{v-1}_{i-\frac{1}{2},j,k} - \delta t\,\delta P^{v}_{i,j,k}/\Delta x_{i-\frac{1}{2}}, \quad (65b)$$

$$v^{\nu}_{i,j+\frac{1}{2},k} = v^{\nu-1}_{i,j+\frac{1}{2},k} + \delta t\, \delta P^{\nu}_{i,j,k} \Big/ \Delta y_{j+\frac{1}{2}}, \tag{65c}$$

$$v^{\nu}_{i,j-\frac{1}{2},k} = v^{\nu-1}_{i,j-\frac{1}{2},k} - \delta t\, \delta P^{\nu}_{i,j,k} \Big/ \Delta y_{j-\frac{1}{2}}, \tag{65d}$$

$$w^{\nu}_{i,j,k+\frac{1}{2}} = w^{\nu-1}_{i,j,k+\frac{1}{2}} + \delta t\, \delta P^{\nu}_{i,j,k} \Big/ \Delta z_{k+\frac{1}{2}}, \tag{65e}$$

and

$$w^{\nu}_{i,j,k-\frac{1}{2}} = w^{\nu-1}_{i,j,k-\frac{1}{2}} - \delta t\, \delta P^{\nu}_{i,j,k} \Big/ \Delta z_{k-\frac{1}{2}}. \tag{65f}$$

This process is continued until the largest value of the divergence, computed from Equation (62), is less than the user specified tolerance. This process can be accelerated when β is multiplied by an over-relaxation parameter, $1.0 \le \omega < 2.0$. Values on the order of 1.8 are often used, but care must be taken not to use too large a value lest the iteration become unstable.[80]

CR Option for the Second Projection Step

The second method present in IPST-VOF3D for accomplishing the correction step in the two-step projection method is the conjugate residual technique. This technique, which requires a symmetric positive definite system of equations for assurance of a solution, is described in detail by Chandra.[90]

Here, The numerical steps needed to solve the system of equations using the CR method are presented. To speed convergence, a preconditioned form of the conjugate residual method is used based on diagonal scaling.[90] Preconditioning modifies the condition number of the system of equations allowing a more rapid solution.

Beginning with a system of equations (in this case the PPE) in matrix form,

$$\mathbf{Ax} = \mathbf{f}, \tag{66}$$

which is assumed to be symmetric and positive definite, the conjugate residual technique seeks to minimize the function

$$E_2(\tilde{\mathbf{x}}) \equiv \left[\mathbf{x}-\tilde{\mathbf{x}}, \mathbf{A}^2(\mathbf{x}-\tilde{\mathbf{x}})\right], \tag{67}$$

where $[\cdot,\cdot]$ denotes an inner product and $\tilde{\mathbf{x}}$ is an approximation of the true solution $\mathbf{x}$ (minimizing $E_1(\tilde{\mathbf{x}}) \equiv \left[\mathbf{x}-\tilde{\mathbf{x}}, \mathbf{A}^1(\mathbf{x}-\tilde{\mathbf{x}})\right]$ yields the more widely known conjugate gradient method). For solution of the Poisson pressure equation, minimizing E_2 is equivalent to driving the divergence towards zero, thereby enforcing the continuity equation.[80]

As mentioned above, the computational technique uses a preconditioned form of the conjugate residual method. First a preconditioning matrix, **M**, which is in some sense close to **A** is chosen. In the case of diagonal scaling, **M** is the diagonal of the coefficient matrix, **A**. Chandra presents the following algorithm for Preconditioned conjugate residual method.[90]

Step 1: Choose $\mathbf{x}_0$

Compute $\mathbf{r}_0 = \mathbf{f} - \mathbf{A}\mathbf{x}_0$

Solve $\mathbf{M}\tilde{\mathbf{r}}_0 = \mathbf{r}_0$

Set $\mathbf{p}_0 = \tilde{\mathbf{r}}_0$

Compute the matrix / vector product $\mathbf{A}\tilde{\mathbf{r}}_0$

Set $\mathbf{A}\mathbf{p}_0 = \mathbf{A}\tilde{\mathbf{r}}_0$ and $i = 0$ (68)

Step 2: Solve the system $\mathbf{M}\mathbf{q}_i = \mathbf{A}\mathbf{p}_i$ (69)

Step 3: Compute

$$a_i = \left[\tilde{\mathbf{r}}_i, \mathbf{A}\tilde{\mathbf{r}}_i\right] / \left[\mathbf{A}\mathbf{p}_i, \mathbf{q}_i\right]$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + a_i\mathbf{p}_i$$

$$\tilde{\mathbf{r}}_{i+1} = \tilde{\mathbf{r}}_i - a_i\mathbf{q}_i$$

$$b_i = \left[\tilde{\mathbf{r}}_{i+1}, \mathbf{A}\tilde{\mathbf{r}}_{i+1}\right] / \left[\tilde{\mathbf{r}}_i, \mathbf{A}\tilde{\mathbf{r}}_i\right] \tag{70}$$

$$\mathbf{p}_{i+1} = \tilde{\mathbf{r}}_{i+1} + b_i\mathbf{p}_i$$

and $$\mathbf{A}\mathbf{p}_{i+1} = \mathbf{A}\tilde{\mathbf{r}}_{i+1} + b_i\mathbf{A}\mathbf{p}_i$$

Step 4: If $\mathbf{x}_{i+1}$ is sufficiently close to $\mathbf{x}$, terminate the iteration process;
Else set $i = i+1$ and go to Step 2. (71)

After the Poisson equation for pressure is solved to yield the new pressure field, the velocity field is updated using a finite difference form of Equation (57):

$$u^{n+1}_{i+\frac{1}{2},j,k} = \tilde{u}^{n}_{i+\frac{1}{2},j,k} - \delta t\left(\delta P^{n+1}_{i+1,j,k} - \delta P^{n+1}_{i,j,k}\right)\Big/\Delta x_{i+\frac{1}{2}}, \tag{72a}$$

$$v^{n+1}_{i,j+\frac{1}{2},k} = \tilde{v}^{n}_{i,j+\frac{1}{2},k} - \delta t\left(\delta P^{n+1}_{i,j+1,k} - \delta P^{n+1}_{i,j,k}\right)\Big/\left(r_i \Delta y_{j+\frac{1}{2}}\right), \tag{72b}$$

and $$w^{n+1}_{i,j,k+\frac{1}{2}} = \tilde{w}^{n}_{i,j,k+\frac{1}{2}} - \delta t\left(\delta P^{n+1}_{i,j,k+1} - \delta P^{n+1}_{i,j,k}\right)\Big/\Delta z_{k+\frac{1}{2}}. \tag{72c}$$

Free Surface Cell Boundary Conditions

Now that I have established a technique for solving the NSE, I turn my attention to the treatment of boundary conditions at the interface between the liquid and vapor phases. First, I present the algorithms used to compute the surface curvature, followed by a description of the velocity boundary conditions near the interface. Finally, I discuss the difficulties in applying the pressure discontinuity imposed by the surface tension force at the interface. In the following section, this discontinuity will be modified to include the deviatoric normal stress in the liquid phase and pressure variations in the vapor phase.

Surface Force Computation

For simplicity I will present the surface curvature finite difference formulas for problems with constant grid spacing. The more complex formulas for variable grids are presented by Torrey et al.[81] and may be deduced from the source code listing presented in Appendix IX. Here, I present only the formulas for Cartesian coordinates (which were not present in the NASA-VOF3D program), Torrey et al.[81] present the cylindrical coordinate formulas.

Only the formula for the surface curvature at an interface with its normal most nearly in the x-direction is presented. In Cartesian coordinates, the y and z direction formulas are available by rotation. The discontinuity in the pressure at the interface due to surface tension, PS, is computed from

$$PS_{i,j,k} = -\sigma\left\{\frac{1}{\Delta y_j}\left[\left(\frac{H_y^x}{\sqrt{1+H_y^{x^2}+H_z^{x^2}}}\right)_{i,j+\frac{1}{2},k} - \left(\frac{H_y^x}{\sqrt{1+H_y^{x^2}+H_z^{x^2}}}\right)_{i,j-\frac{1}{2},k}\right]\right.$$
$$\left. + \frac{1}{\Delta z_k}\left[\left(\frac{H_z^x}{\sqrt{1+H_y^{x^2}+H_z^{x^2}}}\right)_{i,j,k+\frac{1}{2}} - \left(\frac{H_z^x}{\sqrt{1+H_y^{x^2}+H_z^{x^2}}}\right)_{i,j,k-\frac{1}{2}}\right]\right\}, \tag{73}$$

where σ is the surface tension divided by the liquid phase density (remember that the pressure is also divided by the liquid phase density), H_y^x refers to the y-direction derivative of the height function in the x-direction, which is defined as

$$H_{i,j,k}^x = \sum_{i'=i-2}^{i+2}\left[1+AC_{i',j,k}\left(F_{i',j,k}-1\right)\right]\Delta x_{i'}. \tag{74}$$

The partial derivatives needed to compute the surface curvature use nine point computational grids surrounding the surface point of interest. The resulting formulas for the derivatives with respect to the y-direction are

$$H_{y_{i,j+\frac{1}{2},k}}^x = \frac{1}{24\delta y}\left(H_{i,j+1,k+1}^x + 22H_{i,j+1,k}^x + H_{i,j+1,k-1}^x - H_{i,j,k+1}^x - 22H_{i,j,k}^x - H_{i,j,k-1}^x\right), \tag{75a}$$

$$H_{y_{i,j-\frac{1}{2},k}}^x = \frac{1}{24\delta y}\left(H_{i,j,k+1}^x + 22H_{i,j,k}^x + H_{i,j,k-1}^x - H_{i,j-1,k+1}^x - 22H_{i,j-1,k}^x - H_{i,j-1,k-1}^x\right), \tag{75b}$$

$$H_{y_{i,j,k+\frac{1}{2}}}^x = \frac{1}{12\delta y}\left(7H_{i,j+1,k+1}^x - 8H_{i,j,k+1}^x + H_{i,j-1,k+1}^x - H_{i,j+1,k}^x + 8H_{i,j,k}^x - 7H_{i,j-1,k}^x\right), \tag{75c}$$

and

$$H_{y_{i,j,k-\frac{1}{2}}}^x = \frac{1}{12\delta y}\left(7H_{i,j+1,k}^x - 8H_{i,j,k}^x + H_{i,j-1,k}^x - H_{i,j+1,k-1}^x + 8H_{i,j,k-1}^x - 7H_{i,j-1,k-1}^x\right). \tag{75d}$$

Similarly, the derivatives with respect to the z-direction are

$$H^x_{z\,i,j,k+\frac{1}{2}} = \frac{1}{24\delta z}\left(H^x_{i,j+1,k+1} + 22H^x_{i,j,k+1} + H^x_{i,j-1,k+1} - H^x_{i,j+1,k} - 22H^x_{i,j,k} - H^x_{i,j-1,k}\right), \quad (76a)$$

$$H^x_{z\,i,j,k-\frac{1}{2}} = \frac{1}{24\delta z}\left(H^x_{i,j+1,k} + 22H^x_{i,j,k} + H^x_{i,j-1,k} - H^x_{i,j+1,k-1} - 22H^x_{i,j,k-1} - H^x_{i,j-1,k-1}\right), \quad (76b)$$

$$H^x_{z\,i,j+\frac{1}{2},k} = \frac{1}{12\delta z}\left(7H^x_{i,j+1,k+1} - 8H^x_{i,j+1,k} + H^x_{i,j+1,k-1} - H^x_{i,j,k+1} + 8H^x_{i,j,k} - 7H^x_{i,j,k-1}\right), \quad (76c)$$

and

$$H^x_{z\,i,j-\frac{1}{2},k} = \frac{1}{12\,\delta z}\left(7H^x_{i,j,k+1} - 8H^x_{i,j,k} + H^x_{i,j,k-1} - H^x_{i,j-1,k+1} + 8H^x_{i,j-1,k} - 7H^x_{i,j-1,k-1}\right). \quad (76d)$$

Treatment of a dynamic contact line

The formulation presented above for computing the pressure discontinuity due to surface tension is valid so long as the interface between the liquid and vapor phases does not intersect a solid boundary. When this situation occurs, a contact point in two-dimensions or line in three-dimensions occurs. This contact is typically classified as either static or dynamic depending on the motion of the contact point or line relative to the wall (Figure 4 above). The NASA-VOF2D[80] and NASA-VOF3D[81] programs contain limited options for treating dynamic contact points, but make no provision for treating static contact points or lines.

A method to treat dynamic contact points was developed by Torrey et al.[80,81] The terms in Equation (73) of the form

$$\left(H^x_y \Big/ \sqrt{1 + H^{x\,2}_y + H^{x\,2}_z}\right)_{i,j+\frac{1}{2},k} \quad (77)$$

are equivalent to the tension on a discrete "patch" of the interface in the y-direction on the right side of the patch. If the free surface intersects a wall, this tension must be replaced with a wall adhesion force which is equal to the cosine of the user specified contact angle.

Application of the surface pressure as a pressure boundary condition at the interface

So far I have outlined methods for computing the surface pressure discontinuity due to the presence of surface tension or wall adhesion. In the following section, I discuss additions to include the liquid phase deviatoric normal stress and relax the assumption of constant vapor phase pressure. With the surface pressure determined, it is necessary to interpolate (or extrapolate) the surface pressure from the location of the interface to the center of the free surface cell. This is accomplished by making use of a flag array indicating the adjacent computational cell to be used as an interpolation neighbor. The interpolation factor η is computed from the distances shown in Figure 9,

$$\eta = \Delta x_c / \Delta x . \tag{78}$$

The pressure at the center of the free surface cell is then computed from

$$P_{i,j,k} = (1-\eta)P_n + \eta P_s \tag{79}$$

where P_n is the pressure in the neighboring cell and P_s is the pressure at the free surface due to the surface tension or wall adhesion forces. As mentioned above, P_s has been expanded to include the vapor phase pressure and liquid phase deviatoric normal stress.

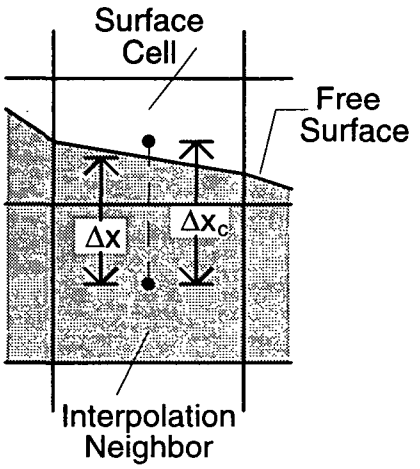

Figure 9. Definition of interpolation geometry.

Solution of the F-Convection Equation

The donor-acceptor algorithm used to solve the F-convection equation is the same as that used in the NASA-VOF2D and NASA-VOF3D computational techniques. First, I state the principles used to derive this algorithm, followed by a discussion of its implementation. More detailed descriptions are presented by Torrey et al.[80,81]

> "The convection algorithms use a form of donor-acceptor differencing which is designed to (1) preserve the sharp definition of free boundaries, which we denote here as fluid interfaces; (2) avoid negative diffusion truncation errors; and (3) not flux more fluid, or more void, across a computing cell interface than the cell losing the flux contains. The algorithms also contain features designed to suppress the appearance of spurious small wisps of fluid."[81]

Generally, the F-convection equation (35) is solved using an explicit finite difference formulation,

$$F_{i,j,k}^{n+\frac{3}{2}} = F_{i,j,k}^{n+\frac{1}{2}} - \frac{\delta t}{AC_{i,j,k}}\left[\frac{r_{i+\frac{1}{2}} AR_{i+\frac{1}{2},j,k} F_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}} u_{i+\frac{1}{2},j,k}^{n+1} - r_{i-\frac{1}{2}} AR_{i-\frac{1}{2},j,k} F_{i-\frac{1}{2},j,k}^{n+\frac{1}{2}} u_{i-\frac{1}{2},j,k}^{n+1}}{r_i \Delta x_i}\right.$$
$$+\frac{ABK_{i,j+\frac{1}{2},k} F_{i,j+\frac{1}{2},k}^{n+\frac{1}{2}} v_{i,j+\frac{1}{2},k}^{n+1} - ABK_{i,j-\frac{1}{2},k} F_{i,j-\frac{1}{2},k}^{n+\frac{1}{2}} v_{i,j-\frac{1}{2},k}^{n+1}}{r_i \Delta y_j} \qquad (80)$$
$$\left.+\frac{AT_{i,j,k+\frac{1}{2}} F_{i,j,k+\frac{1}{2}}^{n+\frac{1}{2}} w_{i,j,k+\frac{1}{2}}^{n+1} - AT_{i,j,k-\frac{1}{2}} F_{i,j,k-\frac{1}{2}}^{n+\frac{1}{2}} w_{i,j,k-\frac{1}{2}}^{n+1}}{\Delta z_k}\right]$$

where the F values are known at the half time steps because of their position in the iterative process. Since for volume of fluid function F is known at the cell centers but is needed at the cell faces in Equation (80), some form of interpolation is required. If a simple average is used, a second order accurate central difference scheme results tending to smear the interface.[80,81]

Therefore, the flux of fluid across each boundary of the computational cell is determined from the donor-acceptor algorithm. First, the donor and acceptor cells, F_D and F_A, respectively, are defined as the upstream and downstream cells based on the velocity in the direction normal to the face of interest. Next, the value of F at the interface, F_{AD}, is chosen to be either F_D or F_A by an algorithm which attempts to impose the first two criteria above. Finally, F_{AD} is modified to impose the third criterion above.

The choice of F_{AD} is made by choosing F_D if the donor cell contains fluid or if the donor cell is nearly tangential to the local surface. If the cell is nearly normal to the interface or the acceptor cell is a void cell, F_A is chosen. This algorithm ensures that the donor cell is nearly full before fluid is convected into an empty acceptor cell and assists in limiting the generation of "spurious wisps" of fluid.[81]

Once F_{AD} has been chosen, corrections to limit the flux are imposed. As an example, the formula at the right face of a computational cell becomes

$$\hat{F}_{AD}V_x = \mathrm{sgn}(V_x)\,\mathrm{MIN}\left(F_{AD}|V_x| + CF,\ F_D\left(x_{ID+\frac{1}{2}} - x_{ID-\frac{1}{2}}\right)\right) \tag{81}$$

where $V_x = u^{n+1}_{i+\frac{1}{2},j,k}\,\delta t$, (82a)

$$CF = \mathrm{MAX}\left[\left(\langle F\rangle - F_{AD}\right)|V_x| - \left(\langle F\rangle - F_D\right)\left(x_{ID+\frac{1}{2}} - x_{ID-\frac{1}{2}}\right), 0.0\right], \tag{82c}$$

and $\langle F\rangle = \mathrm{MAX}(F_D, F_{DM}, 0.1)$ (82b)

with F_{DM} defined as the upstream neighbor to F_D.

The first two options for $\langle F\rangle$ increase the accuracy of the convection process while the third limits convection by the CF term until the donor cell (or its neighbor) has become at lease 10% full. This operation again helps to limit the formation of wisps of fluid. Generally, the MIN statement in Equation (81) ensures that a cell does not give up

more fluid than it contains, and the MAX statement in the definition of CF prevents more void from being convected than is present.

Time Step Limitations

The final step in describing the numerical algorithm used in the IPST-VOF3D computational technique is to discuss the time step limitations. These limitations are general guidelines; cases exist where they are both too restrictive and not restrictive enough. Care should be taken to remember the difference between numerical stability and numerical accuracy. Just because a simulation is numerically stable does not mean that it is an accurate representation of the physical problem.

Next, I describe three primary limiting conditions for the time step; a convective limit, a diffusive limit, and a capillary limit. The absolute convective limit derives from the Courant condition which implies that a particle of fluid may not be convected further than one computational cell during a single time step,

$$\delta t_{conv} < \text{MIN}\left(\frac{\delta x_i}{|u_{i,j,k}|}, \frac{\delta y_i}{|v_{i,j,k}|}, \frac{\delta z_i}{|w_{i,j,k}|} \right), \tag{83}$$

where MIN is the minimum value taken over the entire computational mesh. Typically, a more restrictive limit, CON δt_{conv}, where CON = 0.4 (defined in subroutine **DELTADJ**) is required for numerical stability. This yields an additional constraint on the SOLA upwind differencing parameter, α. If SOLA differencing is used,

$$\alpha > 1.2\,\text{CON} = 0.48 \tag{84}$$

when CON = 0.4.[81]

The diffusive stability limit,

$$\delta t_{visc} = \frac{1}{3\nu\left(\delta x_i^{-2} + \delta y_j^{-2} + \delta z_k^{-2}\right)}, \tag{85}$$

is often the most restrictive condition for problems at low Reynolds numbers.[81] When using the three-dimensional program to solve two-dimensional problems (normally done using a "sandwich" domain with one fluid cell between two slip boundary cells) this limitation can be overly restrictive, so the two dimensional analogue,

$$\delta t_{visc} = \frac{1}{2\nu\left(\delta x_i^{-2} + \delta z_k^{-2}\right)}, \tag{86}$$

may be used (two-dimensional problems are currently only supported in the x-z and r-z planes for Cartesian and cylindrical coordinates, respectively).

The final time step limitation is due to capillary forces. A capillary surface wave must not be allowed to travel through more than one computational cell during a time step. This is approximated by the limitation

$$\delta t_{cap} = \frac{\rho \, \mathrm{MIN}\left(\delta x_i, \delta y_j, \delta z_k\right)}{8\sigma} \tag{87}$$

where, as above, the two-dimensional analogue does not include the y-direction cell size.[81]

MAJOR ADDITIONS TO THE NUMERICAL TECHNIQUE

In addition to the third order accurate techniques for approximating the advective terms in the NSE discussed earlier and the extension of the pressure discontinuity due to surface tension to Cartesian coordinates, there have been several other major additions to the computational technique. These include a method for treating static contact points or lines, inclusion of the liquid phase deviatoric normal stress to the interfacial boundary condition, solution of the potential flow in the vapor phase to relax the assumption of constant vapor phase pressure, and replacement of the graphical output options. These modifications are described in detail in this section.

STATIC CONTACT LINE TREATMENT

Many free surface problems have a contact point or line where the liquid, vapor, and solid phases intersect. As described above, static contact is the intersection between vapor, liquid, and solid phases where the point of contact is fixed relative to the solid surface but the contact angle may vary. The varying contact angle is part of the solution and may have a significant effect on the free surface shape. An example of a problem where this phenomenon is important is the die-swell problem described below.

Previous computational techniques using the VOF approach to track the fluid within a fixed Eulerian mesh[8,80,81] have handled the presence of a dynamic contact line in the manner described in the previous section. That is, the pressure discontinuity at the interface between the liquid and vapor phases in the cell adjacent to the wall was modified to account for the wall adhesion force based on a user specified contact angle. If both the point (or line) of contact and the contact angle are specified in advance, then

the mathematical model of the contact point (or line) is over specified. Thus, either the location or the angle of contact may be specified for a given situation, but not both.

In my treatment of a static contact point (or line), the contact angle is computed from the local fluid configuration and the known contact point (or line). The wall adhesion force needed in the pressure discontinuity is then computed from this variable contact angle in the same manner as for the dynamic contact model described above.

Additional modifications are required in the treatment of the velocity boundary conditions in the region of the static contact point.[40,41] With reference to Figure 10, the velocity boundary condition in the computational cell upstream from the contact point is the same as that for a no-slip wall, i.e. the fictitious velocity inside the die wall is adjusted to enforce zero velocity at the wall. The velocity boundary condition for the downstream computational cell is similar to that of a slip wall, i.e. the velocity inside the wall is set equal to the fluid velocity. Thus, if the explicit projection step for the upstream cell, (i,j), is being computed, the no-slip boundary condition,

$$u_{i+\frac{1}{2},j+1} = -u_{i+\frac{1}{2},j}, \tag{88}$$

is used. If the explicit projection step for the downstream cell, (i+1,j), is being computed then the slip boundary condition,

$$u_{i+\frac{1}{2},j+1} = u_{i+\frac{1}{2},j}, \tag{89}$$

is used. This separate treatment is necessary to reduce the effect of the stress singularity in the interfacial boundary condition at the contact point.

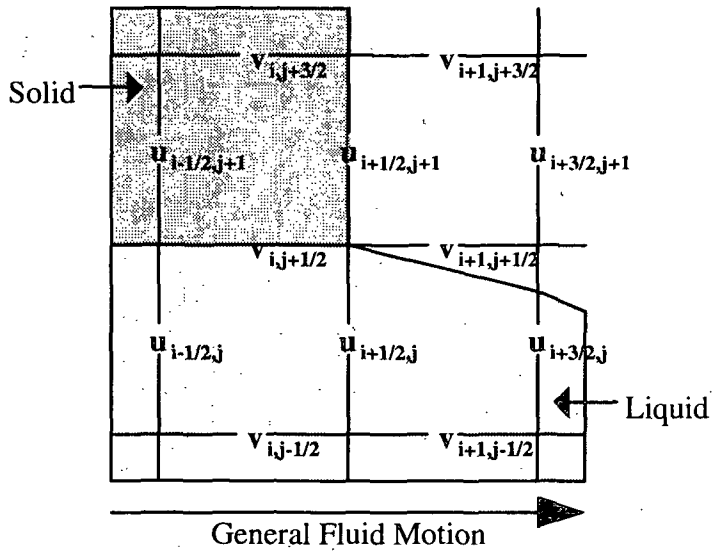

Figure 10. Velocity boundary conditions near a static contact point.

LIQUID PHASE DEVIATORIC NORMAL STRESS

The majority of the volume tracking techniques for free surface flows use Laplace's formula,

$$P_\ell = P_0 + \sigma \kappa, \tag{90}$$

as the interfacial boundary condition arising from the normal stress balance. I have added an option to include the effect of the liquid phase deviatoric stress in the interfacial boundary condition.

The liquid phase deviatoric normal stress at the interface is computed from the complete deviatoric stress tensor and the unit vector normal to the interface, $\mathbf{n} \cdot \tau_\ell \cdot \mathbf{n}$. This requires knowledge of the unit normal vector and all of the components of the deviatoric stress tensor.

The local unit normal vector is computed in the manner used by Torrey et al.[81] during surface tension computations. Once the coordinate axis most nearly normal to the

interface has been determined and the local height functions used to compute the surface curvature have been determined, the unit vector normal to the interface is calculated from the gradient of the height function.

Next, the components of the deviatoric stress tensor are computed using the provisional velocity field, $\tilde{\mathbf{u}}_\ell^n$, where only velocities within the liquid phase are included in the finite difference formulas for the velocity gradients. For example, in two dimensions with reference to Figure 11, the components of the deviatoric stress tensor, assuming constant grid spacing, are computed as:

$$\tau_{xx} = 2\mu \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x}, \tag{91a}$$

$$\tau_{yy} = 2\mu \frac{\left(1+2F_{i,j,k}\right)v_{i,j+\frac{1}{2}} - 4F_{i,j,k}v_{i,j-\frac{1}{2}} - \left(1-2F_{i,j,k}\right)v_{i,j-\frac{3}{2}}}{2\Delta y}, \tag{91b}$$

and

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j} - u_{i+\frac{1}{2},j-1} - u_{i-\frac{1}{2},j-1}}{2\Delta y} + \frac{v_{i+1,j+\frac{1}{2}} - v_{i-1,j+\frac{1}{2}}}{2\Delta x}\right). \tag{91c}$$

The deviatoric stress component most nearly normal to the interface is computed with a finite difference formula that maintains second order accuracy at the location of the interface anywhere within the free surface cell, Equation (91b). The remaining components of the deviatoric stress are computed using finite difference formulas which are at best second order accurate and at worst first order accurate. With the deviatoric stress tensor and the unit normal vector available, the deviatoric normal stress is computed,

$$\text{deviatoric normal stress} = \mathbf{n} \cdot \tau_\ell \cdot \mathbf{n} \tag{92}$$

and used in the free surface boundary condition.

An alternative formulation for the computation of the deviatoric normal stress was developed and is presented in Appendix V. This formulation uses the continuity equation and two zero tangential stress boundary conditions to eliminate three of the components of the deviatoric stress tensor. I chose to use the full implementation because the additional computational cost is small (only a few of the computational cells are free surface cells) and future extension of the computational technique to handle non-Newtonian and turbulent flows is expected to be simpler with the full deviatoric stress.

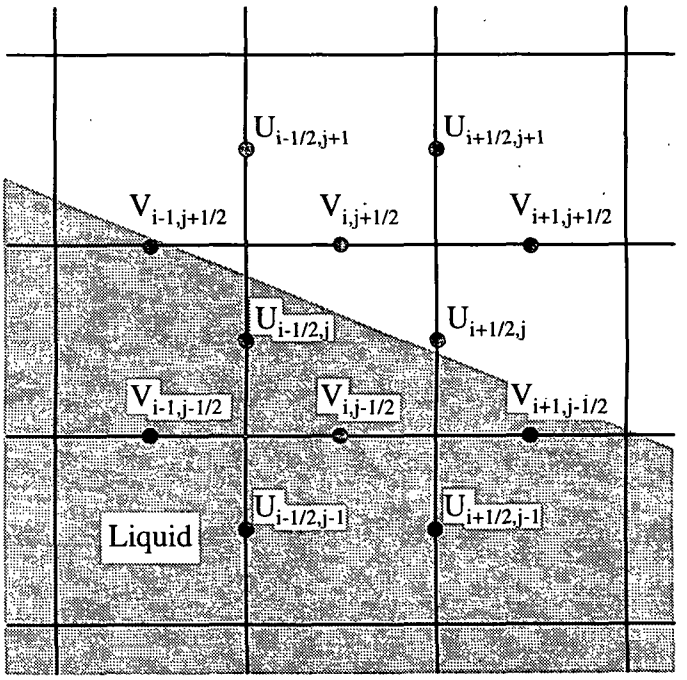

Figure 11. Example configuration for interfacial deviatoric stress computation.

SOLUTION OF POTENTIAL FLOW IN THE VAPOR PHASE

Perhaps the single largest enhancement of the capabilities in the IPST-VOF3D computational technique over previously available VOF-based computational techniques is the relaxation of the constant vapor phase pressure assumption. For many free surface

problems of interest to industry, the physical stability of the interface between the liquid and vapor phases to disturbances is extremely important. In many cases, such as the stability of a thin viscous sheet flowing through an inviscid vapor phase,[58] variations in pressure in the vapor phase is the primary driving force for the instability. Thus, in order to accurately study the stability of these systems, it is necessary to allow the pressure in the vapor phase to vary.

In keeping with the earlier assumption that the vapor phase is inviscid and adding the assumption that flow in the vapor phase is irrotational (this assumption is always valid if the vapor phase is initially stationary and inviscid[91]), then the vapor phase can be represented by potential flow. Thus, the vapor phase potential is defined as,

$$\mathbf{u}_v = \nabla\phi_v, \tag{93}$$

which, after substitution of the vapor phase velocity into the continuity equation, yields Laplace's equation for the vapor phase velocity potential,

$$\nabla^2\phi_v = 0. \tag{94}$$

Under the inviscid and irrotational assumptions when the flow is governed by potential flow. Thus, the unsteady Bernoulli equation,

$$\frac{\partial\phi_v}{\partial t} + \tfrac{1}{2}|\mathbf{u}_v|^2 = -\frac{p_v}{\rho_v}, \tag{95}$$

can be used to relate to vapor phase pressure, vapor phase velocity potential and the vapor phase velocity. After rearranging, the unsteady Bernoulli equation becomes

$$p_v = -\rho_v\frac{\partial\phi_v}{\partial t} - \tfrac{1}{2}|\mathbf{u}_v|^2, \tag{96}$$

which is used to compute the vapor phase pressure as a function of position and time.

The boundary conditions for the vapor phase potential are derived from the vapor phase potential definition in Equation (93). Therefore, the condition at the boundaries of the vapor phase are Neuman (derivative) conditions where the normal derivative of the vapor phase velocity potential is equal to the normal velocity. On the fixed domain boundaries, this condition is easily imposed; however, along the interface between the liquid and vapor phases the Neuman boundary condition is more difficult to apply due to the moving curved boundaries.

A reasonable amount of work has been done previously on methods to accurately treat Neuman or Robbins (mixed constant and derivative) boundary conditions along curved boundaries. Kantorovich and Krylov[92] present a first order accurate method for treating the Neuman boundary condition in two dimensions along a curved boundary. In this formulation, the value of the bulk function, ϕ, at an interface point of interest is defined by

$$\phi_0 = \frac{v_{normal} + \sum_{i=1}^{2} b_i \phi_i}{\sum_{i=1}^{2} b_i}, \tag{97}$$

where v_{normal} is the normal velocity at the interface and two points within the vapor phase are required, one on either side of the normal vector. The coefficients in Equation (97), b_i, are determined by solution of the system of equations,

$$\begin{bmatrix} n_1 & n_2 \\ t_1 & t_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{98}$$

where n_i and t_i in equation (98) are the distances between the surface point and the two points within the domain in the directions normal and tangential to the interface, respectively. Since I will be using a second order accurate approximation of Laplace's

equation for the vapor phase potential in the regions away from the interface and near the fixed boundaries of the computational domain, a second order accurate method for treating the Neuman boundary condition on a curved boundary is needed.

Bramble and Hubbard[93] present a second order accurate technique for solving Poisson's equation in a region with curved boundaries and Robbins boundary conditions. Our problem, solving Laplace's equation in a region with curved boundaries and Neuman boundary conditions, is merely a subset of Bramble and Hubbard's more general problem. They prove that it is possible, within certain constraints surrounding the grid size and local curvature of the interface, to choose three points within the vapor phase which allow computation of positive coefficients. Positive coefficients are necessary to ensure that the resulting system of equations is positive definite and thus, the iterative scheme for solving the system of equations will converge.

The coefficients for the second order accurate formula in two dimensions using three interpolation points is

$$\begin{bmatrix} n_1 & n_2 & n_3 \\ t_1 & t_2 & t_3 \\ t_1^2 - n_1^2 & t_2^2 - n_2^2 & t_3^2 - n_3^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{99}$$

With the proper choice of points to yield positive values for a_i, the value of the vapor phase potential at the interface is defined as

$$\phi_0 = \frac{v_{normal} + \frac{\partial v_{normal}}{\partial t} \sum_{i=1}^{3} a_i n_i t_i + \sum_{i=1}^{3} a_i \phi_i}{\sum_{i=1}^{3} a_i}. \tag{100}$$

A similar technique that is third order accurate was developed by Van Linde[94,95] but is not included in this computational technique. Since the bulk of the liquid phase

differencing is only second order accurate in the spatial derivatives, the extra overhead required for a third order accurate solution of the vapor phase velocity potential does not seem warranted.

Techniques in the literature are available only for two-dimensions. Since I have developed a three-dimensional computational technique for the other aspects of the problem, it was necessary to extend the method of Kantorovich and Krylov[92] and that of Bramble and Hubbard[93] to three dimensions. The details of this process are presented in Appendix VI.

The three-dimensional first order accurate formula derived in Appendix VI is

$$\phi_0 = \frac{v_{normal} + \sum_{i=1}^{3} b_i \phi_i}{\sum_{i=1}^{3} b_i}, \tag{101}$$

where three points within the vapor phase must be chosen and the coefficients are obtained from solution of

$$\begin{bmatrix} n_1 & n_2 & n_3 \\ s_1 & s_2 & s_3 \\ t_1 & t_2 & t_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{102}$$

where n_i, s_i, and t_i are the distances from the surface point to the domain points in the normal and two orthogonal tangential directions, respectively. The only requirement to ensure positive coefficients in the first order accurate case is the three-dimensional analogue of the two-dimensional criterion, that the normal vector must lie within the volume enclosed by the vectors from the surface point to the vapor phase domain points.

The second order accurate formula for three-dimensions is also derived in Appendix VI and represented as

$$\phi_0 = \frac{v_{normal} + \frac{\partial v_{normal}}{\partial s}\sum_{i=1}^{6} a_i n_i s_i + \frac{\partial v_{normal}}{\partial t}\sum_{i=1}^{6} a_i n_i t_i + \sum_{i=1}^{6} a_i \phi_i}{\sum_{i=1}^{6} a_i} \tag{103}$$

where the coefficients are obtained from solution of

$$\begin{bmatrix} n_1 & n_2 & n_3 & n_4 & \bar{n}_5 & n_6 \\ s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ s_1^2 - n_1^2 & s_2^2 - n_2^2 & s_3^2 - n_3^2 & s_4^2 - n_4^2 & s_5^2 - n_5^2 & s_6^2 - n_6^2 \\ t_1^2 - n_1^2 & t_2^2 - n_2^2 & t_3^2 - n_3^2 & t_4^2 - n_4^2 & t_5^2 - n_5^2 & t_6^2 - n_6^2 \\ s_1 t_1 & s_2 t_2 & s_3 t_3 & s_4 t_4 & s_5 t_5 & s_6 t_6 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{104}$$

At this point, no proof has been developed that a collection of points must exist where the coefficients are all positive. However, in practice it has been possible to determine an appropriate set of points in the majority of cases. In the few cases where this has not been possible, the algorithm reverts to the first order accurate formula just described. The fraction of cells where the accuracy of the normal velocity approximation is reduced is generally less than 1% of the interfacial cells and thus is not expected to affect the accuracy of the overall solution. It has also been noted that, for most choices of grid points, the sixth equation can be neglected because the sixth coefficient is nearly zero and approximately the same coefficients are obtained using only the first five equations.

OUTPUT OPTIONS

The choice of graphical and textual output options are controlled by the program input variables LPR, PLTDT, and PRTDT. These and the remaining input variable are

documented along with the program description in Appendix I. Input for LPR has integer values from 0 to 3 yielding options: no output, plots only, plots and prints, and prints only, respectively. When the printing or plotting options are active, PLTDT and PRTDT control the frequency of plotting and printing, respectively.

The printing option creates a text listing of the velocity components, pressure, surface pressure, VOF function (F), cell orientation flags, SOR function (β), interpolation function (η of Equ. 76), and the divergence.

The plotting option has been completely changed from the NASA-VOF2D[80] and NASA-VOF3D[81] programs. These programs contained integrated graphics routines for producing plots of the free surface location, velocity vector, and pressure contour plots. These routines called low level graphics libraries that were specific to the Los Alamos computing environments. I replaced the graphics routines in the earlier programs with a subroutine (**DRAW**) that produces data files suitable as input to the commercial scientific visualization program Data Visualizer™. Here, I present a brief discussion of Data Visualizer's™ wave file format (a more complete description can be found in the Data Visualizer™ programmer's manual[96]).

The input into Data Visualizer™ is in the form of a set of grid points, a topology connecting these grid points, and the data stored at these grid points. There are several different types of topologies possible ranging from structured evenly spaced grids to unstructured grids with extremely complex topology.

The principle output from the **DRAW** subroutine consists of two parts. First a grid data file gives the grid, topology, and information that doesn't vary with time (the volume of each computational cell open to flow). Second, when graphical information is

desired, the simulation data is output including the three components of velocity, the liquid phase pressure, the fraction of each computational cell containing fluid, and the vapor phase potential. In order to produce a Data Visualizer™ input file, the variable data file is appended to the grid file.

Additional graphical output can be obtained by periodically storing subsets of the data in auxiliary storage files and using software such as a spreadsheet to plot the resulting output. This is most easily accomplished through modification of the file *draw.pat*, the appropriate place to modify the **DRAW** subroutine.

The final option for output is based on the restart file. The program, at a user specified interval, saves all of the contents of the common blocks to a data file. This allows the program to be restarted from the saved data if a case needs to be run for a longer time. It also serves as a useful method to "checkpoint" the program. This means that the program can be restarted if it is interrupted during execution (e.g. the computer has been shut down while the program is running). It is possible to write a simple program that reads in a restart file and extracts the necessary data from. This requires a main program that calls the **RDTAPE** subroutine to read in the restart file and then prints the appropriate data.

PRESENTATION OF RESULTS

One of the most important aspects of developing any computational technique is the process of validating the model to ensure that it produces accurate results for test problems where either experimental or other computational results are available. Thus, the presentation of results in my computational work will include several validation problems testing the different features of the computational technique as well as some problems where comparative data may not be available but the results are of interest.

The validation problems include the lid-driven cavity problem for testing the numerical accuracy of the underlying NSE solver; the die-swell problem for testing the free surface capability, the static contact point treatment, and the deviatoric normal stress implementation; and the growth of disturbances in a thin viscous sheet flowing through a stagnant vapor phase. Additional problems studied include the flow under a blade in short dwell coating application[97,98] and rimming flow of condensate in a paper drying cylinder.[99]

In the next several sections, I present the results from each of the test problems just described. Each test problem consists of a description of the problem studied, a brief discussion of results available for comparison, any program modifications and input data necessary to run these cases, and the results themselves.

FLOW IN A TWO-DIMENSIONAL LID-DRIVEN CAVITY

The lid-driven cavity (LDC) problem is a classic test problem for any NSE solution technique. This problem is often used because of its simple geometry having no inflow or outflow boundary conditions with which to contend, while still having a

complex flow field. The problem consists of a box filled with fluid having three stationary walls with the top wall moving (Figure 12).

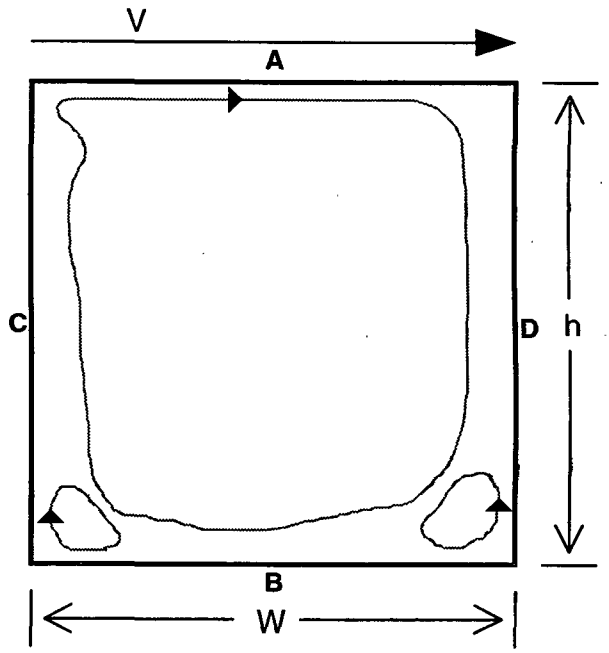

Figure 12. Schematic of the lid-driven cavity.

The dimensionless parameters describing the lid-driven cavity, with reference to Figure 12, are the Reynolds number,

$$Re = hV/\nu, \tag{105}$$

and the cavity aspect ratio,

$$AR = h/W, \tag{106}$$

where h is the height of the cavity, V is the velocity of the cavity lid, ν is the kinematic viscosity of the fluid filling the box, and W is the width of the cavity.

As a test of the accuracy of IPST-VOF3D and especially the third order accurate techniques for the advective terms in the NSE, I studied the lid-driven cavity problem with Re = 1000 and AR = 1. For comparison, the results of Ghia et al.[100] are used which

are among the most accurate available for this highly non-linear problem. Their results were obtained using a multigrid technique on a computational grid with 128 cells in each direction. Our results were obtained on two different computational grids with only 40 computational cells in each direction. The first grid used constant cell spacing while the second was "graded" having smaller cells adjacent to the walls and larger cells in the center of the cavity. The variable grid spacing allows the mesh to be refined in the regions of high gradients, while larger cells are used in regions of lower gradients.

Each of the four differencing techniques for the advective terms in the NSE (SOLA with $\alpha = 0.5$, third order accurate upwind, QUICK differencing, and the method of Kawamura and Kuwahara) were tested using both the constant and variable grids. The results of the horizontal component of velocity along the vertical centerline (line AB in Figure 12) and the vertical component of velocity along the horizontal centerline (line CD in Figure 12) are presented in Figures 13 and 14, respectively. The data for these plots was obtained by reading in the restart files and printing the velocity components along the centerlines as discussed above.

The relative accuracy of the simulations is determined by analysis of the reproduction of the local minimum in Figure 13 and the two local extrema in Figure 14. The average error in reproducing these extrema is presented in Table 2. While using less than 10% the number of computational cells as those used by Ghia et al.,[100] this computational technique was able to get within 5% of their results. All of the third order accurate techniques give acceptable results. Unfortunately, the method of Kawamura and Kuwahara drops from best to worst of the third order accurate methods as the grid is changed from constant to variable. This shift in position is attributed to the inexact nature of the derivation for the variable grid version of Kawamura and Kuwahara's

method. Overall, the third order accurate upwind differencing technique gives the most accurate results for this problem.

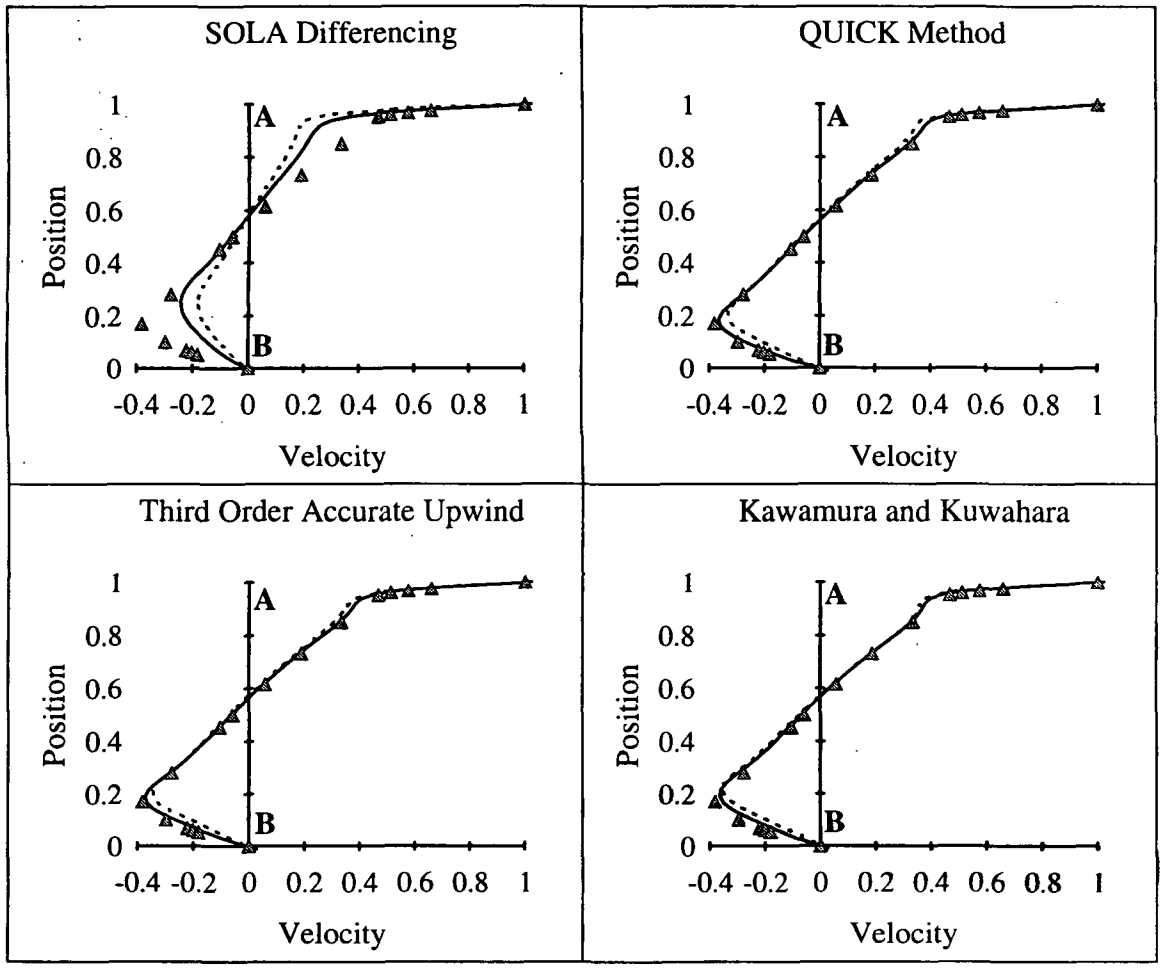

Figure 13. Plots of the horizontal component of velocity along the vertical centerline △ Ghia et al.,[100] ——— variable grid, and - - - - - constant grid.

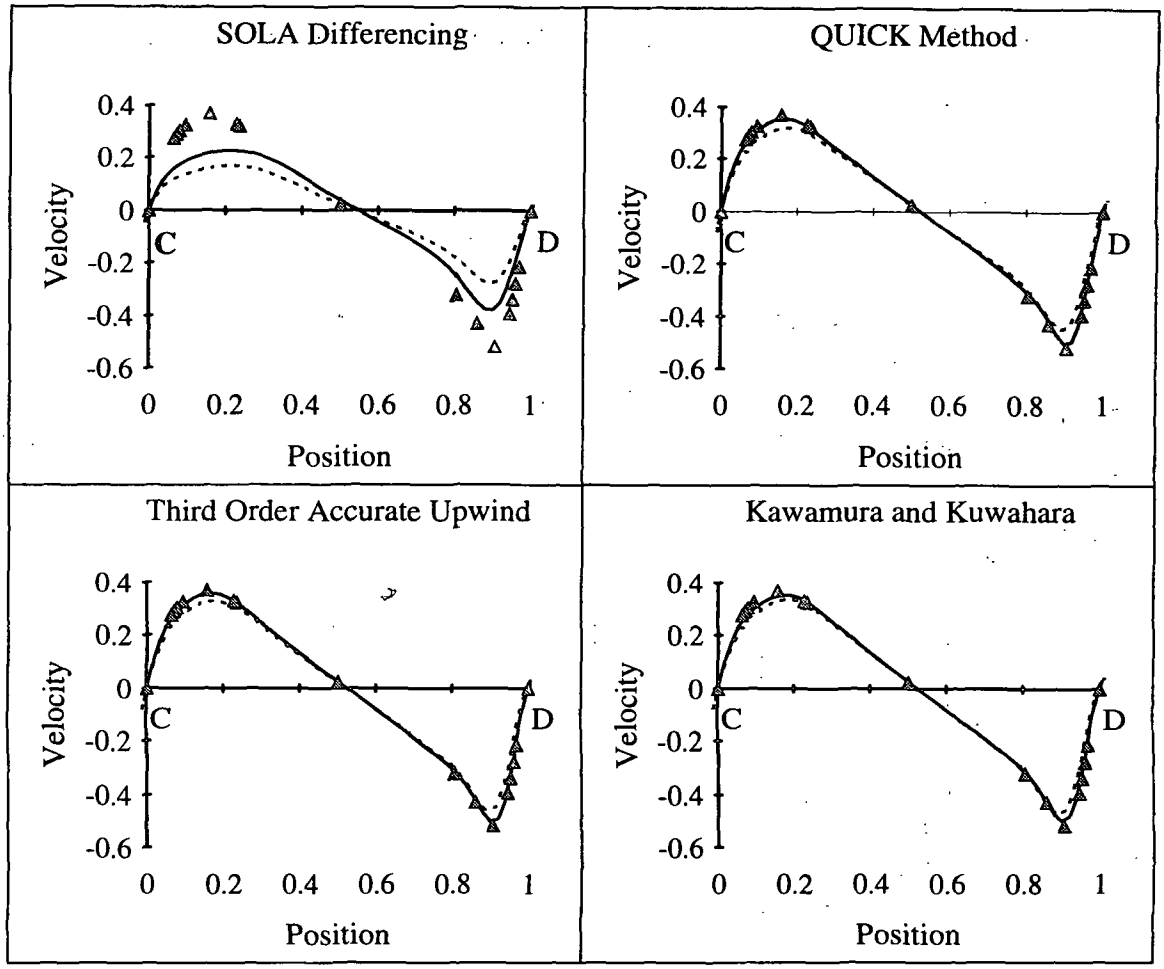

Figure 14. Plots of the vertical component of velocity along the horizontal centerline △ Ghia et al.,[100] ——— variable grid, and - - - - - - constant grid.

Table 2. Error for the LDC problem at Re = 1000.

Convective terms differencing scheme	Grid	
	Constant	Variable
SOLA ($\alpha = 0.5$)	54.3 %	38.8 %
QUICK	15.4 %	4.8 %
Third order accurate upwind	13.0 %	4.2 %
Kawamura and Kuwahara	12.4 %	6.7 %

Using constant grid spacing, Kawai and Ando[101] compared the same three third order techniques with the results from Ghia et al.[100] Their approach used the

streamfunction-vorticity formulation for the NSE and results were obtained on a 60 by 60 computational mesh. Comparison of Kawai and Ando's results for the minimum horizontal velocity component along the vertical centerline with results obtained using IPST-VOF3D is presented in Table 3 with the minimum velocity of –0.3829 obtained by Ghia et al.[100] for comparison.

Table 3. Minimum Velocity along vertical centerline for LDC problem at Re = 1000.

Convective terms differencing scheme	Kawai and Ando	Grid	
		Constant	Variable
QUICK	–0.3680	–0.3340	–0.3676
Third order accurate upwind	–0.3552	–0.3449	–0.3731
Kawamura and Kuwahara	–0.3473	–0.3576	–0.3666

In Kawai and Ando's analysis, QUICK differencing yielded the most accurate result which is inconsistent with the IPST-VOF3D results. In this thesis, third order accurate upwind differencing was the most accurate for the variable grid and more accurate than QUICK using either computational grid. Kawai and Ando also indicate that, for their formulation, QUICK differencing is the only third order scheme tested to remain stable as the Reynolds number is increased above 1000.[101]

The choice of which third order accurate scheme (QUICK differencing or Agarwal's third order accurate upwind differencing) to use for subsequent simulations is by no means clear. Both techniques give comparable accuracy for the lid-driven cavity problem and I have discovered no numerical stability problems with either technique. The majority of the simulations conducted in this thesis have been done using QUICK differencing, primarily due to its reputation for stability and accuracy in the literature.

Problem set up consists of making necessary modifications to the source code of the program and specifying the appropriate input data. For the lid-driven cavity problem, there are no modifications to the source code required and the sample input data for the case with variable grid spacing using SOLA differencing are presented in Table 4. Switching to the third order accurate differencing schemes is accomplished by setting the variable ALPHA in the input data to 2, 3, and 4 for third order accurate upwind, QUICK, and Kawamura and Kuwahara's methods, respectively. The constant grid spacing cases are obtained by modifying the grid setup in Table 4 for the x and z directions to be

```
nkx=1, xl=0.0, 1.0, xc=0.5, nxl=20, nxr=20, dxmn=1.0,
```

and

```
nkz=1, zl=0.0, 1.0, zc=0.5, nzl=20, nzr=20, dzmn=1.0,
```

where the variables are defined as part of the input data in Appendix I and the mechanism for setting up a computational grid was discussed previously.

Table 4. Input data for variable grid LDC with SOLA differencing.

```
$xput
jnm=' vof3d7 ',name=' LDC with SOLA and variable grid', nfcal=1,
iequib=0, icsurf=0, idefm=1, rhof=1.0, cyl=0.0, delt=0.0005,
velmx=1.0, nu=0.001, isor=0, epsi=0.001, wi=0.0,
wl=2, wr=2, wt=2, wb=2, wf=1, wbk=1, utw=1.0,
sigma=0.0, lpr=1, cangle=90.0, isurft=0, alpha=1.0,
dtcrmx=0.05, flht=1.5, twfin=40.0, omg=1.3, pltdt=40.0,
prtdt=40.0, tddt=10.0, tlimd=0.0, td=-1, t=0.0,
$end
$meshgn
nkx=2, xl=0.0, 0.5, 1.0, xc=0.0125, 0.9875,
nxl=1, 19, nxr=19, 1, dxmn=0.0125, 0.0125,
nky=1, yl=0.0, 1.0, yc=1.0, nyl=1,  nyr=0,  dymn=1.0,
nkz=2, zl=0.0, 0.5, 1.0, zc=0.0125, 0.9875,
nzl=1, 19, nzr=19, 1, dzmn=0.0125, 0.0125,
nobs=0,
$end
```

THE DIE-SWELL PHENOMENON

When a fluid is pushed out of the end of die at a low speed it will tend to swell. This die-swell phenomenon is a good test problem for algorithms to numerically study free surface flows. As the Reynolds number of the flow is increased, the amount of swelling decreases until, ultimately, the fluid shrinks (commonly referred to as negative die-swell).

In the die-swell phenomenon, the deviatoric normal stress component of the interfacial boundary condition is extremely important as the fluid near the corner of the die transforms from a no-slip condition inside the die to the zero shear stress condition along the free surface.[40,41] Thus, this problem serves as an excellent test problem for the addition of the liquid phase deviatoric stress to the interfacial boundary condition. The remaining features of the computational technique tested by the die-swell problem are the static contact line treatment and the Cartesian coordinate surface tension algorithm.

Figure 15 shows a sample geometry for the die-swell problem which can be characterized by two dimensionless groups, the Reynolds Number,

$$Re = 2hV/\nu, \tag{107}$$

and the Capillary number

$$Ca = V\mu/\sigma, \tag{108}$$

where h is the slot half height, V is the average velocity of the liquid phase in the slot, ν is the liquid phase kinematic viscosity, μ is the Newtonian viscosity of the liquid phase, and σ is the surface tension between the liquid and vapor phases. The final die swell is defined by

$$\%\ \text{Die-Swell} = 100\left(h_{swell}/h - 1\right). \tag{109}$$

where H_{swell} is the final half thickness of the liquid phase.

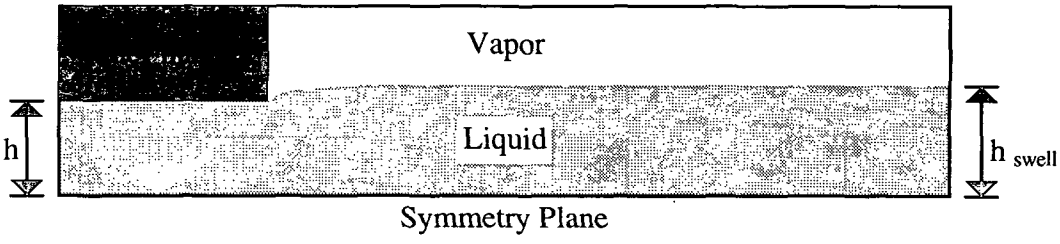

Figure 15. Schematic of the die-swell problem.

Two test cases were studied, the first without surface tension to study the effect of the deviatoric stress alone and the second adds the effect of surface tension. Specifically, the first is Re = 300 and Ca = ∞ while the second is Re = 75 and Ca = 0.5. The static contact line additions are also tested in both cases. The first case examines the effect of the velocity adjustments due to the no-slip to slip transition alone, while the second case tests computation of the wall adhesion force at the static contact point, in addition to the velocity adjustments.

As an example, the input data for the Re = 75, Ca = 0.5 case with domain length of 20h and minimum cell spacing 0.03333h are presented in Table 5. The problem is set up using a symmetry plane and having an entrance region within the die of 3.0h with a parabolic inlet velocity profile. The average inlet velocity, the slot half height, and the viscosity are all set to one, while the remaining parameters of density and surface tension are computed from the Reynolds and capillary numbers. Longer domains are examined by adding additional computational cells to the end of the existing computational domain.

Table 5. Sample input data for the die-swell problem at Re = 75 and Ca = 0.5.

```
$xput
jnm=' vof3d  ',name=' die swell test case 12/20/91 ', nfcal=1,
iequib=0, icsurf=0, idefm=0, rhof=37.5, cyl=0.0, delt=0.005,
velmx=1.0, nu=0.0266666667, isor=0, epsi=0.001, wi=1.0,
sigma=0.0133333333, lpr=1, cangle=90.0, isurft=1, wt=3, wb=3,
alpha=3.0, flht=3.0, twfin=50.0, omg=1.0, pltdt=1.0,
prtdt=1.0, tddt=50.0, tlimd = 0.0, td=-1, t=0.0, dtcrmx=0.005,
```

```
lvapor=0, rhog=0.0001, lvflag=0, islip=0, nfx=1, istress=1,
ifx(1)=32, kfx(1)=31, iorin(1,1)=2, iorin(2,1)=3, islp(1)=0,
$end
$meshgn
nkx=1,xl=0.0,1.3,   xc=1.0,   nxl=30,   nxr=9,    dxmn=1.0,
nky=1,yl=0.0,1.0,   yc=1.0,   nyl=1,    nyr=0,    dymn=1.0,
nkz=1,zl=0.0,20.0,  zc=10.0,  nzl=100,  nzr=100,  dzmn=1.0,
nobs=2,
oa1=-1.0,0.0, oa2=0.0,0.0, ob1=0.0,-1.0, ob2=0.0,0.0,
oc1=1.0,3.0, oc2=0.0,0.0, ioh=1,0,
$end
$fluidgn
$end
```

Modifications to the source code are required in *bc.pat*, *draw.pat*, *setup.pat*, *tilde1.pat*, *tilde2.pat*, and *surcart.pat* presented in Tables 6 through 11, respectively. The modifications in *bc.pat* impose the parabolic profile as an inlet boundary condition and the no-slip conditions along the interior boundaries. The addition in *setup.pat* provides the parabolic velocity profile as an initial condition throughout the entrance region. The changes in *tilde1.pat* and *tilde2.pat* allow the treatment of the upstream and downstream boundary conditions in the region of the static contact point. Finally, *surcart.pat* contains the treatment of the surface tension force at the static contact point.

Table 6. Modifications in *bc.pat* for the die-swell problem.

```
c *** inlet boundary conditions
      do i=1,ifx(1)-1
        u(imax+i)=0.0d0
        w(imax+i)=wi
     1     *1.5d0*(1.0d0-(xi(i)*rx(ifx(1)-1))**2)
        w(imax+i+ii5)=w(imax+i)
      enddo
c *** die-edge boundary conditions
      do k=1,kfx(1)-1
        ijk=ii5*(k-1)+imax+ifx(1)
        w(ijk)=-w(ijk-1)*delx(ijk)*rdx(ijk-1)
        u(ijk-1)=0.0d0
      enddo
      kkk=ii5*(kfx(1)-1)+imax
      do i=ifx(1),im1
        u(kkk+i)=u(kkk+i+ii5)
        w(kkk+i)=w(kkk+i+ii5)
```

```
      enddo
      u(ii5*(kfx(1)-1)+imax+ifx(1)-1)=0.0d0
ccc   end addition
```

Table 7. Modifications in *draw.pat* for the die-swell problem.

```
ccc   addition to print out surface profile
      write(2,'(/,'' time='', 1pe12.4)') t
      do k=2,kfx(1)
        write(2,'(i6,3f15.10)') k, zk(k), x(ifx(1)-1), x(ifx(2))
      enddo
      do k=kfx(1)+1,km1
        temp4=0.0d0
        do i=2,im1
          ijk=imax+ii5*(k-1)+i
          if (nf(ijk).ge.1.and.nf(ijk).le.6) then
            temp4=x(i-3)+delx(i-2)*f(ijk-2)+delx(i-1)*f(ijk-1)
     1        +delx(i)*f(ijk)+delx(i+1)*f(ijk+1)+delx(i+2)*f(ijk+2)
            go to 2121
          endif
        enddo
 2121   continue
        write(2,'(i6,3f15.10)') k, zk(k), temp4, temp2
      enddo
ccc   end addition
```

Table 8. Modifications in *setup.pat* for the die-swell problem.

```
ccc  additon for parabolic initial condition
      if (beta(ijk+ii5).gt.0.0d0.and.beta(ijk+ii5).ne.1.0d0)
     1  w(ijk)=wi *1.5d0*(1.0d0-(xi(i)*rx(ifx(1)-1))**2)
ccc  end additon
```

Table 9. Modifications in *tilde1.pat* for the die-swell problem.

```
ccc   addition for singularity at the static contact points
      do itemp1 = 1, nfx
        if (iorin(1,itemp1).le.2) then
          itemp2=-3+2*iorin(1,itemp1)
          itemp3=-3+iorin(2,itemp1)
          if (i.eq.ifx(itemp1)-itemp2.and.
     1      k.eq.kfx(itemp1)-itemp3) then
            temp1=wn(ijk+itemp2)
            wn(ijk+itemp2)=wn(ijk)
            if (islp(itemp1).eq.0.and.(islip.eq.0.or.
     1        (nf(ijk+itemp3*ii2).eq.0.and.islip.eq.1)))
     2        wn(ijk+itemp2)=-wn(ijk)
          endif
        else
```

```
            itemp3=-7+2*iorin(1,itemp1)
            itemp2=-1+iorin(2,itemp1)
            if (i.eq.ifx(itemp1)-itemp2.and.
     1        k.eq.kfx(itemp1)-itemp3) then
              temp1=un(ijk+itemp3*ii2)
              un(ijk+itemp3*ii2)=un(ijk)
              if (islp(itemp1).eq.0.and.(islip.eq.0.or.
     1          (nf(ijk+itemp2).eq.0.and.islip.eq.0)))
     2          un(ijk+itemp3*ii2)=-un(ijk)
            endif
          endif
        enddo
ccc     end addition
```

Table 10. Modifications in *tilde2.pat* for the die-swell problem.

```
ccc     addition for singularity at the static contact points
        do itemp1=1,nfx
          if (iorin(1,itemp1).le.2) then
            itemp2=-3+2*iorin(1,itemp1)
            if (i.eq.ifx(itemp1)-itemp2.and.
     1        k.eq.kfx(itemp1)+3-iorin(2,itemp1)) wn(ijk+itemp2)=temp1
          else
            itemp2=-7+2*iorin(1,itemp1)
            if (i.eq.ifx(itemp1)+1-iorin(2,itemp1).and.
     1        k.eq.kfx(itemp1)-itemp2) un(ijk+itemp2*ii2)=temp1
          endif
        enddo
ccc     end addition
```

Table 11. Modifications in *surcart.pat* for the die-swell problem.

```
c +++ addition for static contact point
        if (i.eq.ifx(1)-1.and.k.eq.kfx(1)+1) then
          if (nf(ijk).eq.1.or.nf(ijk).eq.2) then
            afs=0.0d0
          else if (nf(ijk).eq.5.or.nf(ijk).eq.6) then
            afe=0.0d0
          endif
          if (f(ijk).lt.0.5d0) then
            temp1=2.0d0*f(ijk)*delz(k)*rdx(i)
            csang=temp1/dsqrt(1.0d0+temp1*temp1)
          else
            temp2=2.0d0*(1.0d0-f(ijk))*delx(i)*rdz(k)
            csang=1.0d0/dsqrt(1.0d0+temp2*temp2)
          endif
        endif
c +++ end addition
```

Die-Swell with Re = 300 and Ca = ∞

The results from the die-swell problem at Re = 300, Ca = ∞ are presented in Table 12, where the computed die-swell ratio is recorded as a function of the length of the computational domain and the size of the computational cells adjacent to the static contact point. Since the deviatoric normal stress acting on the free surface is greatest near the static contact point, the finest computational mesh is required in this region. The number of computational cells remains constant while the minimum cell spacing is adjusted in the computational cell nearest to the die corner. This leads to the smallest cells near the die and largest cells near the symmetry plane. Generally, the die-swell ratio decreases as the minimum cell spacing decreases or the computational domain length increases. An asymptotic value of −15.66% is reached, in good agreement with literature results of −15.52% and −15.24% reported by Omodei[68] and FIDAP,[71] respectively.

The actual surface profiles from several of the cases in Table 12 are presented in Figure 16 where each chart consists of the surface profiles for the separate cases at a given cell spacing. Only every tenth data point is presented to simplify the plots. The initial portions of each figure are in good agreement, while small waves in the interfacial position are seen as the domain length is increased.

Table 12. Results of solutions of the die-swell problem at Re = 300, Ca = ∞.

		Minimum Cell Spacing			
		0.04h	0.03h	0.02h	0.01h
Domain Length	20h	−14.48%	−14.92%	−15.20%	−15.12%
	25h	−14.78%	−15.08%	−15.52%	−15.37%
	30h	−15.05%	−15.14%	−15.62%	−15.53%
	35h	−15.19%	−15.17%	−15.66%	−15.63%
	40h	−15.31%	−15.21%	−15.66%	−15.67%

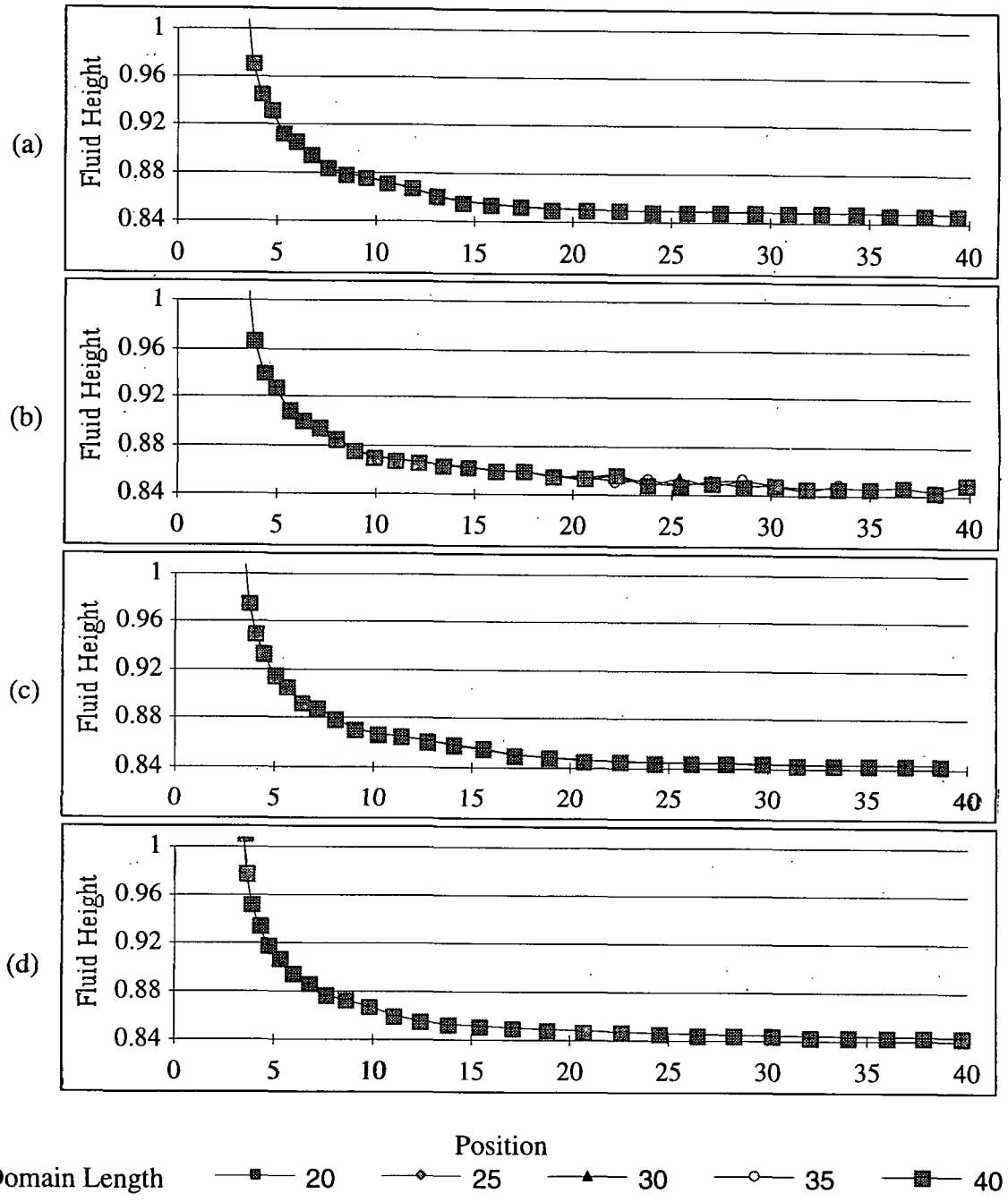

Figure 16. Surface profiles for domain lengths of 20H, 25H, 30H, 35H, and 40H with minimum cell spacings of (a) 0.04H, (b) 0.03H, (c) 0.02H, and (d) 0.01H.

Die-Swell with Re = 75 and Ca = 0.5

The die-swell problem with surface tension was studied in a slightly different manner. Here, the length of the computational domain was held constant at 20H while the number of computational cells perpendicular to the direction of flow and the minimum cell spacing adjacent to the static contact point were varied. The results of these simulations are presented in Table 13. Note that certain combinations of cell spacing and number of computational cells are not possible as indicated. This results because not that many cells with the minimum cell spacing will fit within the computational domain (e.g. 36 cells with a cell spading of 0.0333H yields 1.2H for the die half height which is greater than H). The results for this test problem show a good deal more scatter, but a majority of the cases fall within the range of the literature values which are −11.16%,[71] −10.92%,[102] and −10.48%.[68] The surface profiles for the cases along the diagonal of Table 13 are presented in Figure 17.

Table 13. Results of solutions of the die-swell problem at Re = 75 and Ca = 0.5.

		Minimum Cell Spacing			
		0.0333h	0.0278h	0.0222h	0.0167h
Y-Direction Computational Cells	30	−10.99%	−11.61%	−11.38%	−11.26%
	36	***	−11.62%	−11.43%	−11.53%
	45	***	***	−10.91%	−11.37%
	60	***	***	***	−11.05%

*** combination not possible

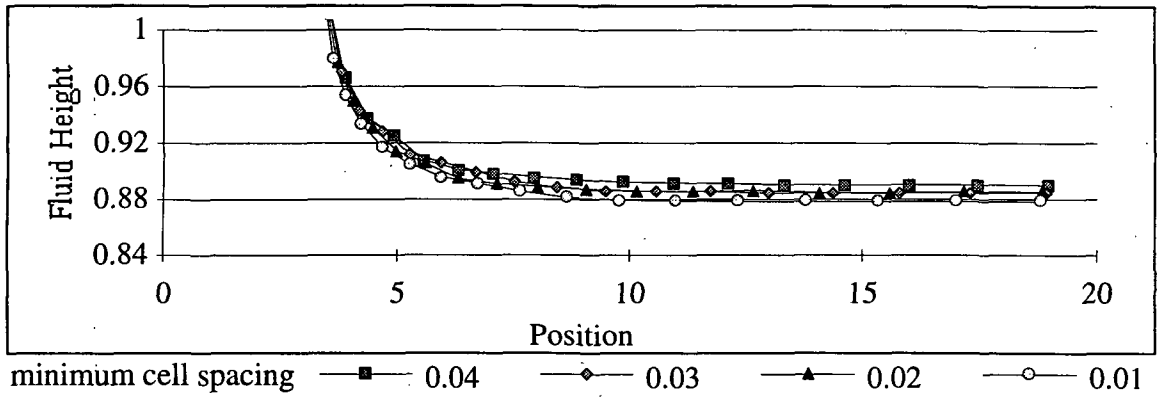

Figure 17. Surface profiles for RE = 75 and Ca = 0.5 (a) 0.0333H with 30 cells, (b) 0.0278H with 36 cells, (c) 0.0222H with 45, and (d) 0.0167H with 60.

The Effect of the Deviatoric Stress in the Interfacial Boundary Condition

The importance of including the deviatoric stress in the interfacial boundary condition for accurate solution of the die-swell problem has been discussed previously. Here, simulations of the two cases discussed above solved with and without the inclusion of the liquid phase deviatoric stress in the interfacial condition are presented.

Figures 18 and 19 show the effect of the deviatoric stress on specific cases outlined in the Tables above. As expected, at the higher Reynolds number, Figure 18, the effect of the deviatoric stress is less important than at the lower Reynolds number, Figure 19. Regardless of the Reynolds number, the inclusion of the deviatoric stress in the interfacial condition is necessary for accurate solution of the die-swell problem.

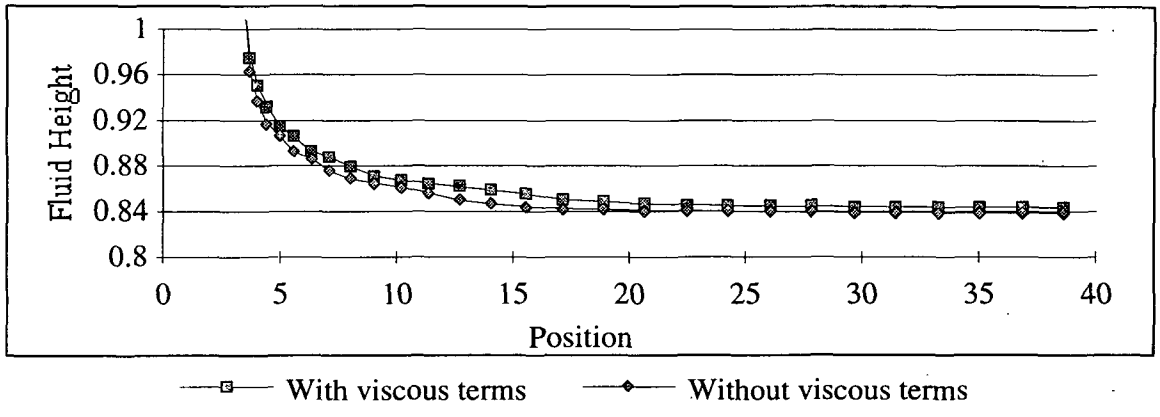

Figure 18. The effect of the viscous terms in the interfacial boundary condition for the Re = 300, Ca = ∞ case having a minimum cell spacing of 0.02 and a domain length of 40.

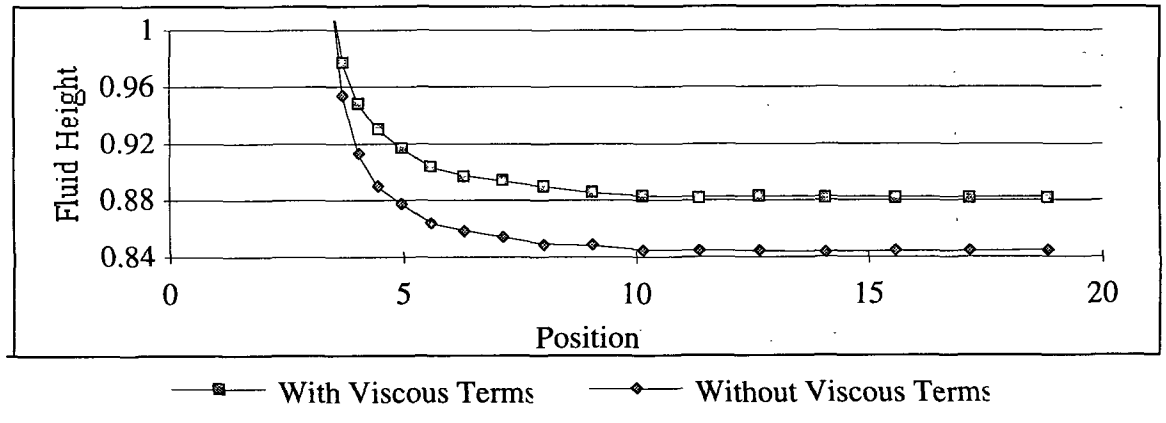

Figure 19. The effect of the viscous terms in the interfacial boundary condition for the Re = 75, Ca = 0.5 case having a minimum cell spacing of 0.02 and 50 computational cells in the fluid half-height.

In summary, these results indicate that accurate solution of the die-swell problem is possible provided the liquid phase deviatoric normal stress is included in the interfacial boundary condition and there is proper treatment of the surface tension and velocity boundary condition at the edge of the die. The deviatoric stress in the interfacial boundary condition becomes increasingly important as the Reynolds number decreases.

STABILITY OF A THIN TWO-DIMENSIONAL VISCOUS SHEET

As discussed above, the stability of a thin viscous sheet of fluid flowing through a stagnant inviscid vapor phase is the problem driving the requirements for this computational technique. Thus, the final test problem for the computational technique is the accurate prediction of the growth rates of waves in the viscous sheet. In order for accurate solution of this problem to be possible, all of the added features of the computational technique must be working properly and accurately with the exception of the static contact line treatment which is not tested by this problem.

Simulations at $We_\ell = 40$, $Z = 0.1$, and $\tilde{\rho} = 0.1$ have been conducted for both antisymmetric and axisymmetric disturbances at wavenumbers of 1, 2, and 3 with both the top and the bottom surfaces simulated and including both vapor phase regions. Solutions at m = 1 were obtained on a computational domain with $2\pi a$ in the primary direction of flow and 8a perpendicular to the flow. This problem was discretized on a computational grid with 360 constantly spaced cells in the direction of flow and 100 graded cells perpendicular to the primary direction of flow. The grading was done so that regions of constant cell spacing were maintained adjacent to the interfaces. Problems with larger wavenumbers used the same computational grid with shorter computational domains to keep the number of computational cells per wavelength constant. A computational domain with periodic boundary conditions was used allowing the traveling waves to leave the domain while reentering it from the opposite side. The input data for the antisymmetric case with m = 1 are presented in Table 14. The code modifications needed are the imposition of the initial surface perturbation and the surface position output. These modifications, shown in Tables 15 and 16 are included of the **SETVEL** subroutine and *draw.pat*, respectively.

Table 14. Input data for antisymmetric wave growth problem with m = 1.

```
$xput
name=' antisymmetric wave growth m = 1 - Z=0.1, We=4.0, Rv/Rl=0.1',
rhof=63.24555321d0, rhog=0.1d0, sigma=0.025d0, nu=0.01581139d0,
wi=1.0d0,
jnm=' vof3d7 ', nfcal=1, iequib=0, icsurf=1, idefm=1, cyl=0.0d0,
delt=0.0001d0, velmx=1.0d0, isor=1, epsi=1.0d-4, lpr=1, cangle=90.0d0,
isurft=1, wt=4, wb=4, alpha=3.0d0, flht=0.0d0, twfin=30.0d0,
omg=1.75d0,
pltdt=0.5d0, prtdt=1.0d0, tddt=30.0d0, tlimd=0.0d0, td=-1, t=0.0d0,
dtcrmx=0.005d0, islip=1, nfx=0, epsiv=1.0d-4, autot=1,
ifx(1)=72, kfx(1)=2, ifx(2)=31, kfx(2)=2,
istress=1, lvapor=1, lvflag=1, vomg=1.65d0,
nvfr=2, ivfr=2,101, jvfr=2,2, kvfr=2,2, ivwl=1, ivwr=1, ivwb=2, ivwt=2,
$end
$meshgn
nkx=4, xl=0.d0, 5.01d0, 6.d0, 6.99d0, 12.d0,
xc=4.61d0, 5.41d0, 6.59d0, 7.39d0,
nxl=20, 10, 10, 10, nxr=10, 10, 10, 20,
dxmn=0.02d0, 0.02d0, 0.02d0, 0.02d0,
nky=1, yl=0.0d0, 1.0d0, yc=1.0d0, nyl=1, nyr=0, dymn=1.0d0,
nkz=1, zl=0.0d0, 6.283185307d0, zc=3.141592654d0, nzl=160, nzr=160,
dzmn=1.0d0,
$end
$fluidgn
nqbs=2,
qa1=1.0d0, 1.0d0, qa2=0.0d0, 0.0d0, qb1=0.0d0, 0.0d0, qb2=0.0d0, 0.0d0,
qc1=-7.0d0, -5.0d0, qc2=0.0d0, 0.0d0, iqh=1, 0,
$end
```

Table 15. Initial perturbation for antisymmetric wave growth with m = 1 in *setvel.f*.

```
      subroutine setvel
      include 'vof3dcom'
c
      integer k, kkk
      real*8 mm, kk, temp1, aaa
c
      mm=1.0d0
      aaa=0.5d0*(xi(ifx(1))-xi(ifx(2)))
      kk=mm/aaa
c
      do k=2,km1
        kkk=ii5*(k-1)+imax
        temp1=zeta*rdx(ifx(1))*dsin(kk*zk(k))
        f(kkk+ifx(1))=f(kkk+ifx(1))+temp1
        f(kkk+ifx(2))=f(kkk+ifx(2))-temp1
      enddo
```

```
c
      call dcopy(lvec, f, 1, fn, 1)
      call dcopy(lvec, fn, 1, fnn, 1)
c
      return
      end
```

Table 16. Modifications to *draw.pat* for the wave growth problem.

```
c
      write(2,290) t
      do k=2,km1
        do i=imax/2,2,-1
          ijk=imax+ii5*(k-1)+i
          if (nf(ijk).ge.1.and.nf(ijk).le.6) then
            temp2=x(i+2)-(delx(i-2)*f(ijk-2)+delx(i-1)*f(ijk-1)
     1        +delx(i)*f(ijk)
     2        +delx(i+1)*f(ijk+1)+delx(i+2)*f(ijk+2))
            go to 1212
          endif
        enddo
 1212   continue
        do i=imax/2+1,im1
          ijk=imax+ii5*(k-1)+i
          if (nf(ijk).ge.1.and.nf(ijk).le.6) then
            temp4=x(i-3)+(delx(i-2)*f(ijk-2)+delx(i-1)*f(ijk-1)
     1        +delx(i)*f(ijk)
     2        +delx(i+1)*f(ijk+1)+delx(i+2)*f(ijk+2))
            go to 2121
          endif
        enddo
 2121   continue
        write(2,300) k, zk(k), temp4, temp2
      enddo
c
  290 format (/,' time=', 1pe12.4)
  300 format (i6,3f15.10)
c
```

Figure 20 presents a plot of the Ln(amplitude) as a function of time for a specific case as an example. The displacement is computed as the average of the displacement of one crest and one trough on each interface, where the same crests and troughs are followed as they move through the computational domain. Since from linear stability analysis the growth is expected to be exponential, the plot in Figure 20 is expected to be a straight line, which after an initial transient region, it is. The dimensionless growth rate

determined from the slope of the linear portion of this curve combined with the slopes for additional cases at wavenumbers 1 and 3 and axisymmetric cases at wavenumbers 1, 2, and 3 are presented in Figure 21. These results show very good agreement between the theoretical and computed growth rates in the linear growth rate regime.

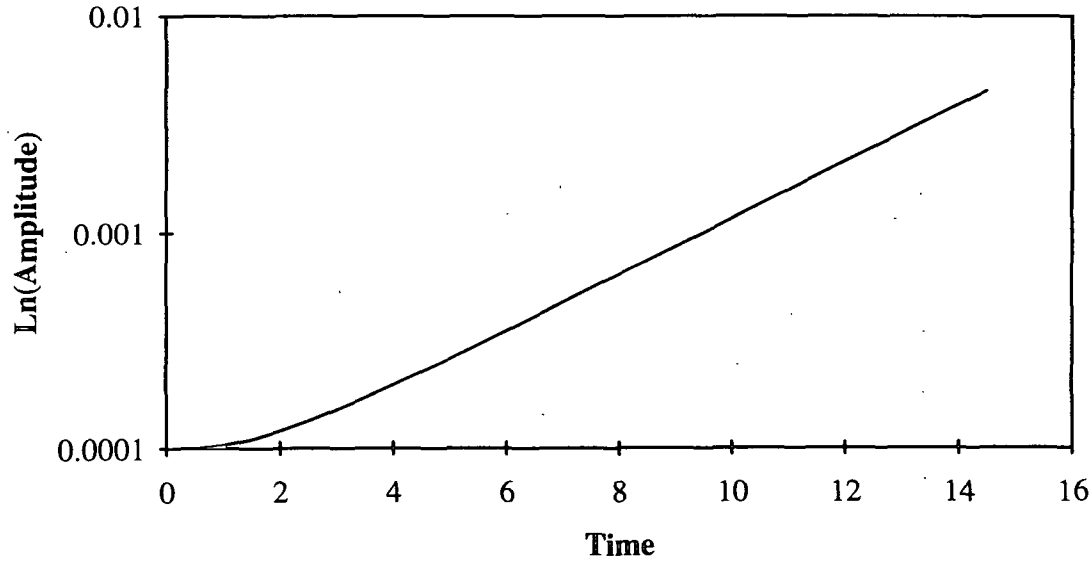

Figure 20. Wave growth results for antisymmetric case for $We_\ell = 40$, $Z = 0.1$, $\tilde{\rho} = 0.1$, and $m = 2$.

The results presented in Figure 21 imply that, given sufficient grid resolution, the physics associated with the treatment of boundary conditions at the interface between a liquid phase and a vapor phase governed by potential flow are reasonably treated.

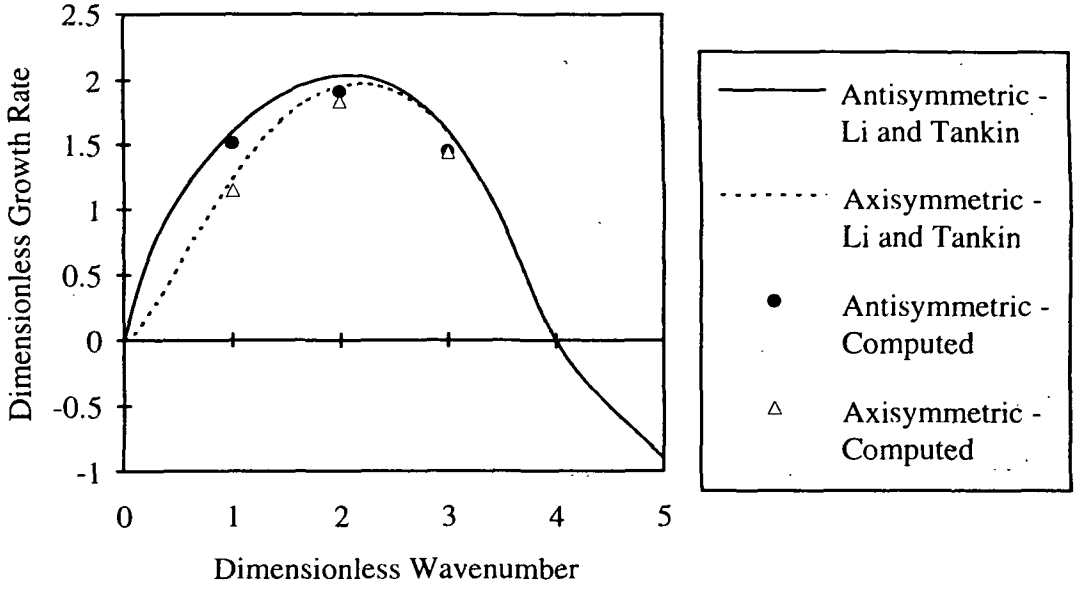

Figure 21. Non-dimensional growth rate for $We_\ell = 40$, $Z = 0.1$, and $\tilde{\rho} = 0.1$ obtained from numerical solution of the Li and Tankin's[58] dispersion relations. Closed circles and open triangles represent results from computational analysis.

Thus, with the solution of the lid driven cavity problem, the die-swell problem, and the accurate prediction of the growth of waves in a thin viscous sheet of fluid, I have demonstrated the accuracy of the treatment of the interfacial boundary conditions. I now go on to discuss applications of this computational tool to specific problems that are of interest to the pulp and paper industry.

THE EFFECT OF CROSS MACHINE DIRECTION PRESSURE VARIATIONS ON COAT-WEIGHT NON-UNIFORMITIES

Non-uniformities in the coated surface of paper is a problem that becomes more pronounced as the speed of coating operations increases. Specifically, in short dwell time coaters, an uneven coat-weight profile appears characterized by streaks of 1 to 3 cm wide running along the machine direction (MD). In general, these streaks are approximately 15 to 50 % thinner than other coated regions of the sheet, and their scale, ~1 cm, is much larger than the blade gap (~30-50 μm). The magnitude of these streaks is in contrast to other coating defects such as streaks due to solid particles blocking the blade gap or skip coating where the streaks have essentially no coating.[97]

In addition, wet streaks occur when the coating speed is increased above a critical limit for a given coating formulation. Pilot trials by Triantafillopoulos and Aidun[103] and Li[104] indicate that the limiting speed decreases with increasing percentage solids and, consequently, the low-shear viscosity of the coating color.[97]

There have been three primary mechanisms (Figure 22) proposed for the formation of wet streaks.[97] The first is air entrainment at the dynamic contact line[105] which occurs as the contact line becomes unstable. The second and third proposed mechanisms are due to hydrodynamic instability in the flow within the pond of the short dwell coater. Instability in these three-dimensional flows can lead to cross-machine direction pressure variations in the coating entering the blade gap. Finally, the presence of the deformable blade and substrate can allow for variation in the blade gap and, thus, the coating thickness.

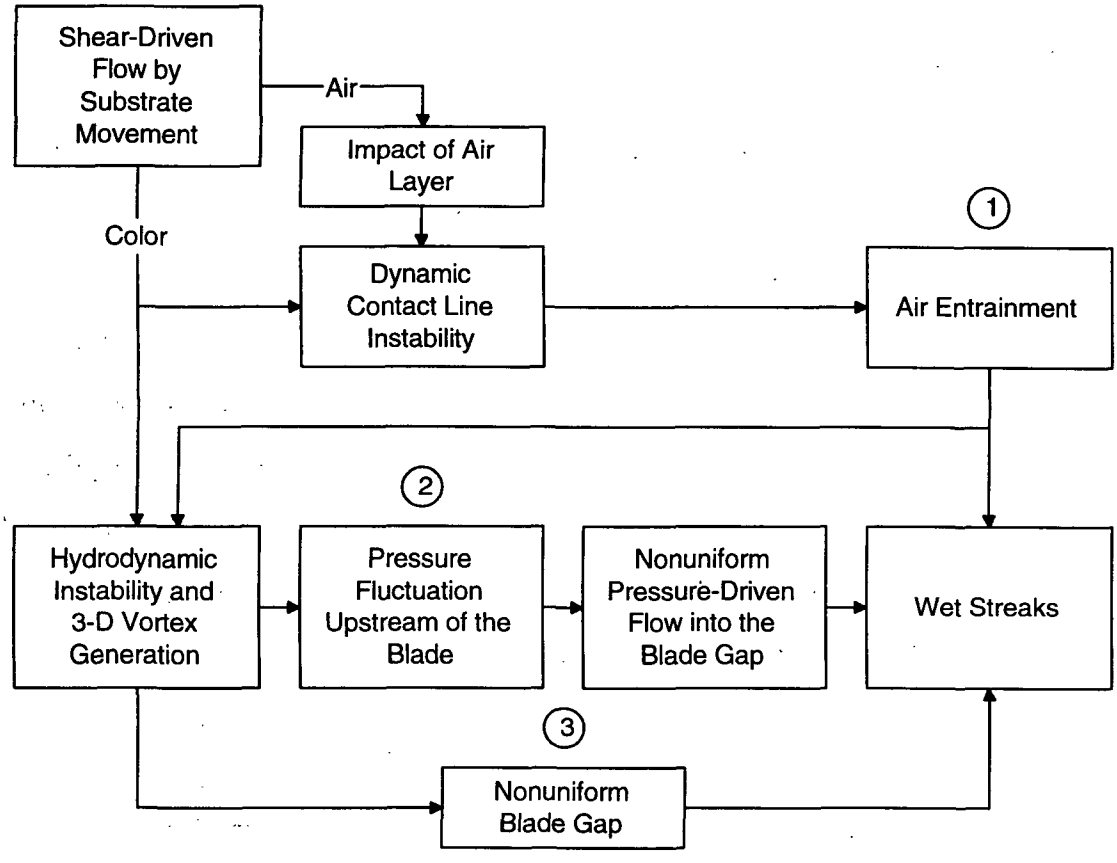

Figure 22. Summary of mechanisms which may lead to wet streaks.[97]

In this thesis, results from study of the second mechanism, the effect of pressure variations upstream of the blade on the thickness of the coating layer in the absence of the air entrainment and blade gap fluctuation mechanisms are presented.

Here, I discuss the results obtained by Miura and Aidun,[97] using an early version of the IPST-VOF3D program, in their study of the effects of pressure variations upstream of the blade on the thickness of the coating leaving the blade region. First, I present their results from the study of the temporal pressure variations. This is followed by an extension of Miura and Aidun's spatial fluctuation results.[98]

Two-Dimensional Base Case

The initial condition for all of the cases discussed below is a two-dimensional steady-state simulation. Since the computational technique developed in this dissertation is transient, the normal method for obtaining a steady state solution is to choose a reasonable initial condition and march forward in time until the solution ceases to vary with time. The results of the steady-state two-dimensional simulation yield a better initial guess for the three-dimensional simulations performed below, potentially reducing the time needed to reach steady-state. The 2D-SS problem solved is shown in Figure 23. All of the results are presented in nondimensional form with the blade gap and the substrate velocity used as characteristic length and velocity, respectively.

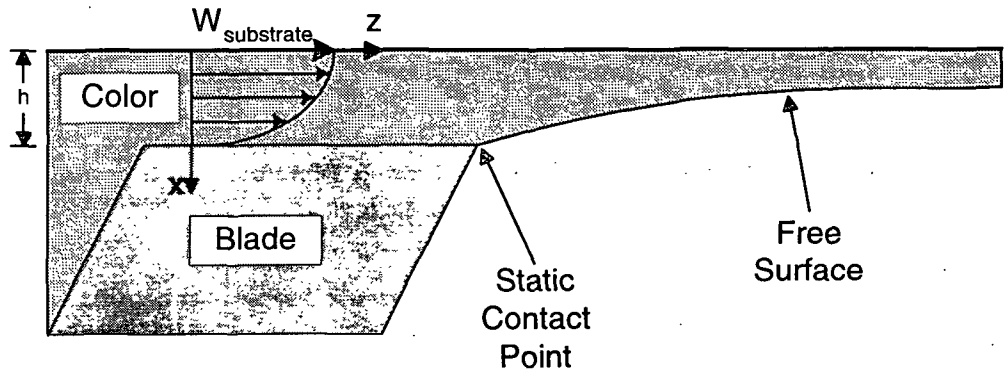

Figure 23. Schematic of the two-dimensional steady-state coating problem.

The inlet boundary condition assumes that the fluid is adjacent to the upstream edge of the blade and the flow is a linear combination of shear-driven (Couette-flow) and pressure-driven (Poiseuille-flow).[97,106,107] Thus, the nondimensional inlet velocity profile is given as

$$w(x,y,t) = 1 - x + \frac{1}{2}\frac{dP}{dz}(y,t)\,x(x-1) \quad \text{at} \qquad z = 0 \tag{110}$$

where the pressure gradient, $\frac{dP}{dz}(y,t)$, will later be allowed to vary with span wise position, y, and time, t. Results are to be presented in terms of the Reynolds number,

$$Re = \rho h W_{substrate}/\mu, \tag{111}$$

and Capillary number,

$$Ca = \mu W_{substrate}/\sigma, \tag{112}$$

where $W_{substrate}$ is the web speed, ρ is the density of the coating color, μ is the viscosity (assumed to be constant with a value appropriate for the conditions under the blade),[97] h is the blade gap, and σ is the surface tension. The details of the two-dimensional steady-state simulations and demonstration of the solution grid independence are presented by Miura and Aidun[97] and Miura et al.[98]

Example input data for the two-dimensional steady-state simulation are presented in Table 17. The input data are entered in dimensional units in the mks system. The code modifications required are the same as those for the die-swell problem discussed above except that different velocity profiles are required for the inlet and initial conditions. The temporal instability problem discussed here and the three-dimensional problem discussed below require adjustments in *bc.pat* and *setup.pat* for the velocity profiles which are presented in Tables 18 and 19, respectively. The modifications for the treatment of the static contact point and the display of the interface profile are accomplished in the same manner as those used for the die-swell problem discussed above. Notice that temporal and spatial fluctuations in the inlet pressure gradient can be imposed by adjusting the comments in front of specific lines in *bc.pat* presented in Table 18.

Table 17. Input data for the two-dimensional steady state simulation of flow under a short dwell coater blade.

```
$xput
jnm=' QUICK ',name='50 cps  03/20/92 ', nfcal=2,
iequib=0, icsurf=0, idefm=1, rhof=1.2, cyl=0.0, delt=0.000000001,
velmx=1000.0, nu=0.4167, isor=0, epsi=0.005, wi=0.0, wlw=1000.0,
sigma=41.67, lpr=1, cangle=90.0, isurft=1, autot=1.0, wl=2, wt=3,
wb=3, wf=1, wbk=1, alpha=3.0, dtcrmx=0.0000005, flht=0.1,
twfin=0.0005, omg=1.0, pltdt=0.0001, prtdt=0.0001, tddt=0.0005,
tlimd=0.0, td=-1, t=0.0, islip=0, nfx=1, ifx(1)=20, kfx(1)=31,
iorin(1,1)=2, iorin(2,1)=3, islp(1)=0, istress=1,
$end
$meshgn
nkx=1, xl=0.0, 0.005555555556, xc=0.005, nxl=18, nxr=2,  dxmn=1.0,
nky=1, yl=0.0, 1.0, yc=1.0, nyl=1,  nyr=0,  dymn=1.0,
nkz=1, zl=0.0, 0.4, zc=0.1, nzl=30, nzr=90, dzmn=0.0005,
nobs=2,
oa1=-1.0, 0.0, oa2=0.0, 0.0, ob1=0.0, -1.0, ob2=0.0, 0.0,
oc1=0.005, 0.1, oc2=0.0, 0.0, ioh=1, 0,
$end
$fluidgn
$end
```

Table 18. Modifications to *bc.pat* for the flow under a short dwell blade.

```
c *** blade-edge boundary conditions
      do j=2,jm1
        jjj=imax*(j-1)
        do k=1,kfx(1)-1
          ijk=ii5*(k-1)+jjj+ifx(1)
          w(ijk)=-w(ijk-1)*delx(ijk)*rdx(ijk-1)
          u(ijk-1)=0.0d0
        enddo
        kkk=ii5*(kfx(1)-1)+jjj
        do i=ifx(1),im1
          u(kkk+i)=u(kkk+i+ii5)
          w(kkk+i)=w(kkk+i+ii5)
        enddo
        u(ii5*(kfx(1)-1)+jjj+ifx(1)-1)=0.0d0
      enddo
c
c     set specified inflow velocities for designated inflow cells
c
      do j=2,jm1
        jjj=imax*(j-1)
        do i=2,ifx(1)-1
          ijk=jjj+i
            u(ijk) = 0.0
```

```
            v(ijk) = 0.0
c    --- parabolic profile ---
c           w(ijk)=wi*1.5*(1.0-(xi(i)*rx(ifx(1)-1))**2)
c    --- inflow setting : couette & poiseuille flow ---
c    ---   temp1 is pressure gradient  : dp/dz (dyn/cm^2/cm)
            temp1 = -1.60e8
c    --- time periodic pressure fluctuation
c    ---   prtdt is the period of the fluctuation
c           if (t.le.2.0d0*prtdt) then
c             temp2=1.0d0+0.5d0*dsin(2.0d0*pi*t/prtdt)
c    --- CD pressure fluctuation
c    --- full-wave fluctuation ? ---
c             temp2=1.0d0+0.5d0*dcos(2.0d0*pi*yj(j)/yl(2))
c    --- or half-wave fluctuation ? ---
c             temp2=1.0d0+0.5d0*dcos(1.0d0*pi*yj(j)/yl(2))
c    ------------------------------------------------
            w(ijk) = wwl*(1-xi(i)*rx(ifx(1)-1))+
     1       temp1*temp2/(2.0*nu*rhof)*xi(i)*(xi(i)-x(ifx(1)-1))
c    ------------------------------------------------
          endif
        enddo
      enddo
```

Table 19. Modifications to *setup.pat* for the flow under a short dwell blade.

```
c --- modified for blade coating simulation  by Miura 06/04/91 ---
c    --- inflow setting : couette & poiseuille flow ---
c          -1.6e8 is pressure gradient in md : dp/dz (dyn/cm^2/cm)
                w(ijk) = wwl*(1.0d0-xi(i)*rx(ifx(1)-1))+
     1           (-1.60e8)/(2.0*nu*rhof)*xi(i)*(xi(i)-x(ifx(1)-1))
```

Results from the two-dimensional steady state base case problem are presented in Figure 25 for the coarse and fine computational grids diagrammed in Figure 24.

The slight discontinuities or "kinks" in Figure 23 result from a combination of three factors. First, the position of the interface is computed using only a local height function in the y-direction. As the computational grid is refined, even though it remains skewed, the magnitude to the kinks decreases and the surface becomes smoother, in part because the y-direction height function becomes more accurate. The second factor contributing to the kinks is the highly skewed grid structure which reduces the accuracy

of the surface shape computation. For example in the coarse grid simulation the y-direction grid spacing is 0.1 while the x-direction grid spacing is 0.25 yielding a grid aspect ratio of 2.5 to 1 versus the ideal 1 to 1 ratio. The final factor affecting the rough appearance of the interface is the scaling of the y-axis, which is magnified over 7 times contributing to the apparently large size of these kinks.

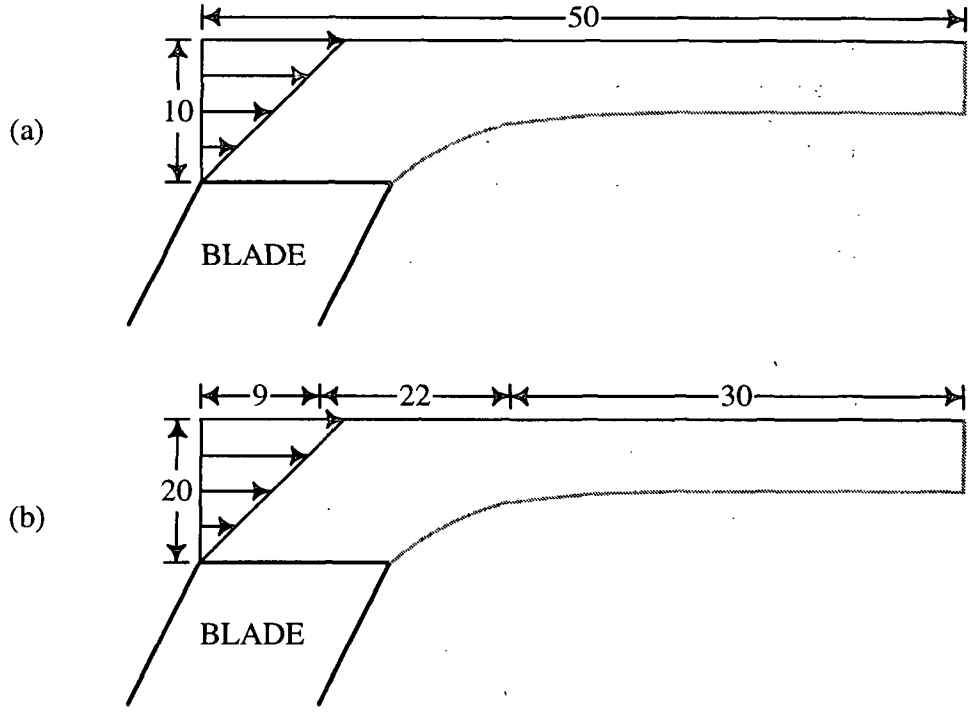

Figure 24. Illustration of the (a) coarse and (b) fine grid systems for the two-dimensional computations (number inside the arrow indicates the number of cells).

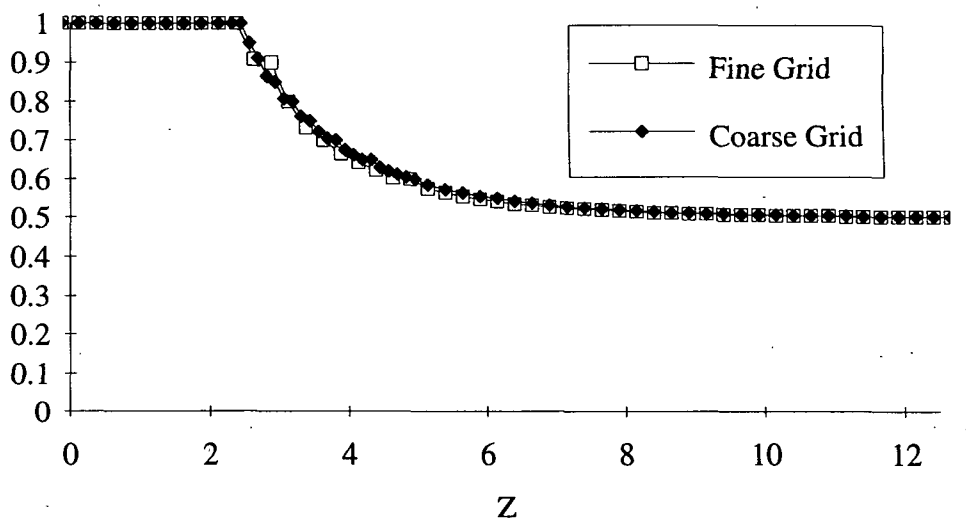

Figure 25. Comparison of the free surface computations using the fine and coarse grid systems.

Next, I focus on the results of Miura and Aidun's simulations. First, I discuss their study of time periodic pressure fluctuations in two-dimensions, followed by presentation and extension of three-dimensional steady-state simulations with span wise pressure variations.

Time-Periodic Pressure Fluctuation

In addition to wet streaks, which are coat-weight non-uniformities in the span wise direction, the film thickness can also develop a 2-D wavy profile due to temporal fluctuation in the pressure upstream of the blade. Thus, the effect of the time-dependent fluctuation of static pressure at the blade entrance on the film thickness was investigated.

A sinusoidal pressure fluctuation lasting for two periods is given by

$$\frac{dP}{dz}(t) = \left\langle \frac{dP}{dz} \right\rangle \left[1 + 0.5\sin(2\pi t/T)\right], \qquad 0 < t < 2T$$
$$\text{and} \quad \frac{dP}{dz}(t) = \left\langle \frac{dP}{dz} \right\rangle, \qquad t \geq 2T \tag{113}$$

where $\left\langle \frac{dP}{dz} \right\rangle$ indicates the steady-state pressure gradient and T is the period of fluctuation. Notice that the amplitude of the pressure varies by ±50% from the steady-state value.

With the addition of the time periodic inflow condition, all other boundary conditions remain the same as the steady-state simulation discussed above, which was used as the initial condition. The blade gap was maintained at 50 mm and the substrate speed varied from 10 m/s for Re = 12 to 25 m/s for Re = 30. The flow parameters in dimensional and dimensionless form are presented in Table 20.

Table 20. Physical parameters used in temporal pressure variation study.

Parameter	Dimensional	Dimensionless
Blade gap, h	5×10^{-5} m	1
Blade thickness, τ	1×10^{-3} m	20
Density, ρ	1200 kg/m^3	—
Viscosity, ν	0.05 Pa·s	—
Machine Speed, W Average Pressure Grad., $\left\langle \frac{dP}{dz} \right\rangle$	10 m/s -100 MPa/m	1 -0.5 (Re = 12)
W $\left\langle \frac{dP}{dz} \right\rangle$	15 m/s -150 MPa/m	1 -0.5 (Re = 18)
W $\left\langle \frac{dP}{dz} \right\rangle$	20 m/s -200 MPa/m	1 -0.5 (Re = 24)
W $\left\langle \frac{dP}{dz} \right\rangle$	25 m/s -250 MPa/m	1 -0.5 (Re = 30)

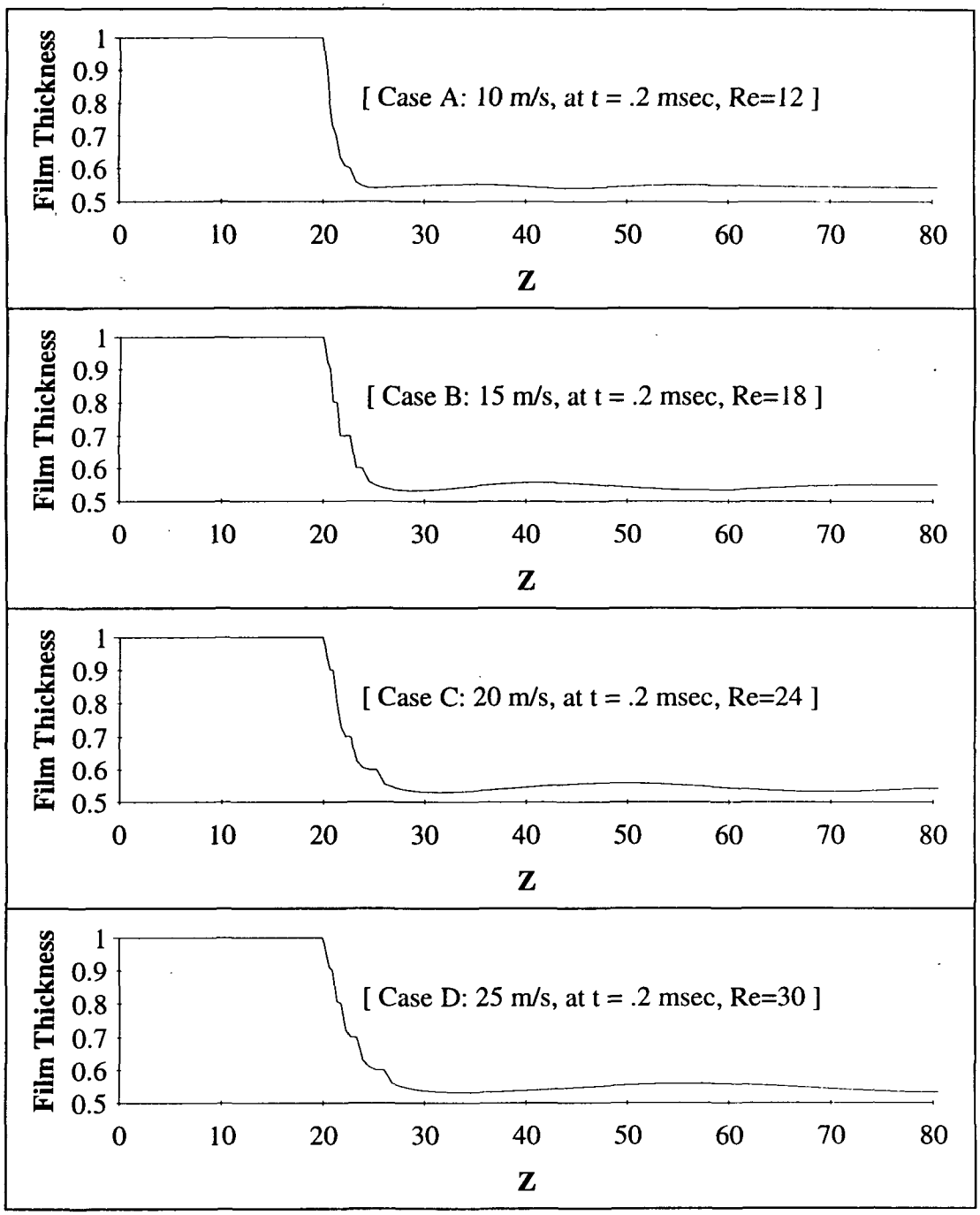

Figure 26. Variation in surface thickness due to temporal pressure fluctuation at the blade entrance after two periods of variation.

Figure 26 shows the film thickness as a function of position after two periods of disturbance. The variations in thickness are small, which is to be expected since more than 91.7% of the mass flow through the gap in this case is shear driven.[97] Thus, a 50% variation in pressure could be expected to yield ~4% variation in film thickness. This film thickness variation from the steady-state value is plotted in Figure 27. Notice that the computed film thickness variations are less than 1%, much smaller than the 4% variation expected from the simple linear analysis.

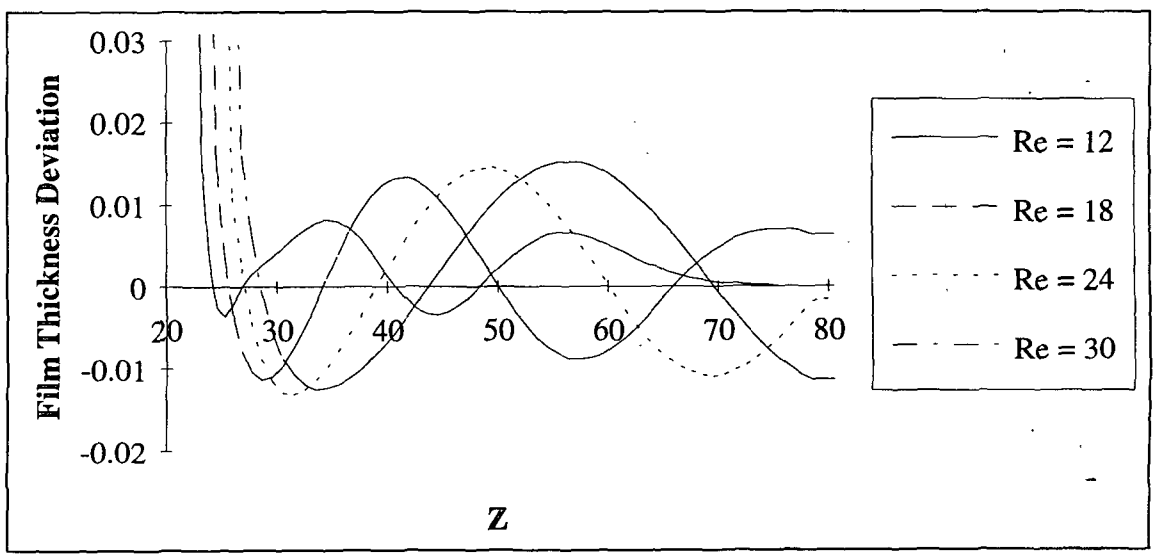

Figure 27. Film thickness deviation from the steady state outlet value.

Cross-Machine Pressure Fluctuation

Here we study the effects of cross machine pressure variations on the steady-state coating profile downstream of the blade. Various types of flow instability in the pond are expected to yield pressure variations with wavelengths from 1 mm to greater than 1 cm.[97]

Film thickness non-uniformities become even more important in light weight coated applications where the blade gap may be reduced to 30 μm from the 50 μm studied in the temporal instability. The dimensional and dimensionless flow parameters used in the three-dimensional simulations are presented in Table 21 where the Reynolds and capillary numbers fall within the range of actual operating conditions. Paper coating fluids are typically shear thinning, but for these simulations a constant viscosity with the value expected in the shear rates under the blade is used. The pressure gradient is estimated from the results of Pranckh and Scriven[106] which is currently the most complete two-dimensional analysis of blade coating.

The pressure variation is assumed to be a sinusoidal disturbance given by

$$\frac{dP}{dz}(y) = \left|\frac{dP}{dz}\right|\left[1 + 0.5\cos(\pi y/y)\right] \qquad (114)$$

where $\left|\frac{dP}{dz}\right|$ is the average pressure gradient and y is the pressure fluctuation wavelength. The boundary conditions remain the same as in the previous simulations with the exception of the inlet condition and the additional constraints of symmetry conditions,

$$v = \frac{\partial u}{\partial y} = \frac{\partial w}{\partial y} = \frac{\partial P}{\partial y} = 0 \qquad \text{at} \qquad y = \pm y/2. \qquad (115)$$

Miura and Aidun[97] studied the effect of dimensionless span wise wavelengths from 33.3 to 400, corresponding to dimensional disturbances from 1 mm to 1.2 cm. Here, this analysis has been extended to dimensionless wavelengths of 800 and 1200 corresponding to dimensional disturbances of 2.4 cm and 3.6 cm, respectively.

Table 21. Physical parameters used in three-dimensional spatial pressure variation study.

Parameter	Dimensional	Dimensionless
Substrate speed, W	20 m/s	1
Blade gap, h	3×10^{-5} m	1
Blade thickness, τ	1×10^{-3} m	20
Density, ρ	1200 kg/m^3	—
Viscosity, ν	0.05 Pa·s	—
Surface Tension, σ	0.05 N/m	—
Average Pressure Grad., $\left\|\frac{dP}{dz}\right\|$	-1.6×10^9 Pa/m	-1.44
Reynolds number, Re	—	14.4
Capillary number, Ca	—	20

With no CD pressure variations, the steady-state film thickness can be computed by integrating Equation (110). This indicates that the average contributions to the dimensionless film thickness from Couette-flow and Poiseuille-flow are

$$\underset{\text{Couette}-\text{flow}}{0.5} + \underset{\text{Poiseuille}-\text{flow}}{0.12} = \underset{\text{Total}}{0.62} \tag{116}$$

A simple linear analysis predicts that a CD pressure fluctuation of 50 % will lead to 50 % Poiseuille flow fluctuations and dimensionless film thickness variations from 0.56 to 0.68. As can be seen in Figure 28, when the wavelength of the disturbance is large, the amplitude coat weight variations agrees with the simple linear analysis, but when the wavelength is very small, the amplitude of the surface profile is also very small.

Miura and Aidun[97] present the following discussion of the physical mechanism leading to the variation in film thickness as a function of disturbance wavelength:

> Considering the y-component of the Navier-Stokes Equation and noting that the length and velocity scale in the x-direction is small, that is $O(\delta)$, it is easy to show from order of magnitude analysis that the pressure gradient term is balanced mainly by
>
> $$\partial p/\partial y \cong \partial^2 v/\partial x^2 .$$
>
> When we compute the RHS term from the computational results, we observe that its magnitude does not greatly vary between the cases

considered in this study. Therefore, the value of $\partial p/\partial y$ is also of the same order for y between 1 mm and 12 mm. This implies that as the wavelength of the pressure fluctuation, i.e., the magnitude of the denominator, ∂y, increases, the magnitude of the pressure gradient, i.e., the magnitude of the numerator, ∂p, will increase to keep $\partial p/\partial y$ relatively constant. As we mentioned above, however, at a critical wavelength the value of the pressure gradient will approach a plateau with a further increase in y.

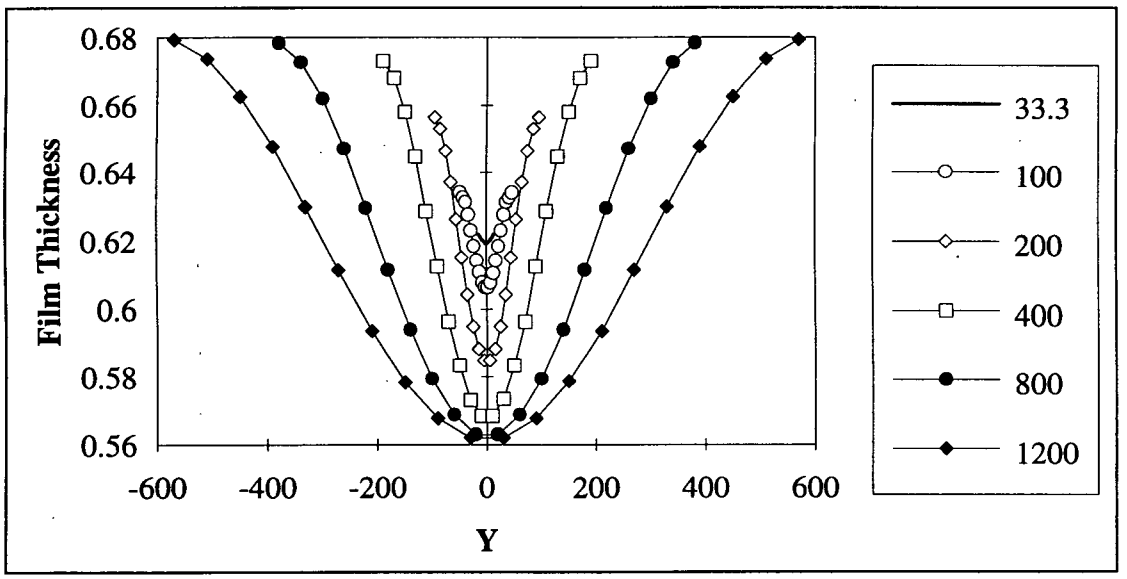

Figure 28. Comparison of the span wise film thickness profile for pressure variations having varying wavelength. Cases with wavelength of 33.3, 100, 200, and 400 by Miura and Aidun.[97]

These results and analysis indicate that the second proposed mechanism in Figure 22, that of pressure fluctuations upstream of the blade, is indeed a plausible mechanism for wet streaks. For the case studied, a 50% cross machine direction pressure fluctuation on the order of 1 to 3 cm in side yields a coat weight variation on the order of 10%. Disturbances with smaller cross machine fluctuations yield smaller coat weight variations which may not be perceived as wet streaks.

This is not expected to be the complete story of the wet streak phenomenon. This analysis does not include the compressibility of the substrate or the deflection of the blade, two results that are also expected to contribute to coat weight variations via a non-uniform blade gap (mechanism 3 in Figure 22).

CONDENSATE FLOW INSIDE DRYER CYLINDERS

In this section, results obtained by applying this computational technique to the study of condensate flow within dryer cylinders are discussed. In this problem, a constant volume of fluid is contained in a rotating horizontal cylinder. No attempt was made to include the effects of steam condensation, variation of properties with temperature, or the siphoning process used to remove condensate from the cylinder. The focus of this study was to determine the velocity profiles within the condensate layer and to understand the implications of these profiles on the heat transfer through the condensate layer.

A schematic of the condensate flow problem is presented in Figure 29. It is shown as a two-dimensional problem, although a three-dimensional computational domain was required for solution. Simplifications are present in the IPST-VOF3D program to improve the computational efficiency in the study of two-dimensional problems; however, it is currently only possible to treat two-dimensional problems in the x-z or r-z planes in this manner. Since this two-dimensional problem is defined in the r-θ plane, a domain with two fluid cells and two fictitious cells in the z-direction was used, combined with symmetry boundary conditions in the z-direction to simulate a two-dimensional computational domain.

The condensate flow problem is characterized by the dimensionless groups Reynolds number,

$$Re = \rho\omega b^2/\mu, \tag{110}$$

(sometimes defined as $Re = \rho\omega R^2/\mu$ in the literature); the Froude number,

$$Fr = \omega^2 R/g; \tag{111}$$

the Capillary number,

$$Ca = \mu\omega R/\sigma; \tag{112}$$

and the film thickness ratio,

$$\delta = b/R. \tag{113}$$

Here, R is the radius of the inside wall of the cylinder, ω is the angular velocity of the rotating cylinder, b is the average condensate film thickness, g is the acceleration due to gravity (acting downward in Figure 29), ρ is the condensate density, μ is the condensate viscosity, σ is surface tension of the condensate.

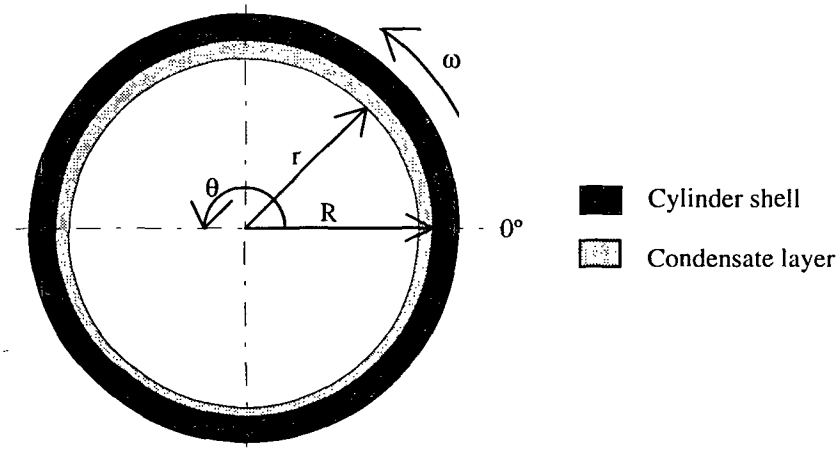

Figure 29. Schematic of the condensate flow problem.

Earlier studies of this problem have been conducted by Deibler and Cerro[108] as well as by Orr and Scriven.[109] Deibler and Cerro used an order of magnitude analysis to generate a simplified form of the NSE for this problem,

$$V_r \frac{\partial V_\theta}{\partial \Re} + \frac{V_\theta}{\Re}\frac{\partial V_r}{\partial \theta} = -\frac{\cos\theta}{Fr} + \frac{\delta^2}{Re}\frac{\partial}{\partial \Re}\left[\frac{1}{\Re}\frac{\partial(\Re v_\theta)}{\partial \Re}\right], \tag{114}$$

where $V_r = v_r/\omega R$, $V_\theta = v_\theta/\omega R$, and $\Re = r/R$. This equation, essentially a boundary layer equation in cylindrical coordinates, was solved numerically in transformed

coordinates for Reynolds numbers from 0.05 to 456, for Froude numbers from 0.1 to 16, and for film thickness ratios from 0.05 to 0.0004.

Orr and Scriven[109] studied the rimming flow problem using a finite element technique to solve the full incompressible NSE for flow in the liquid layer. In comparing their results, it is important to note that they have defined the Reynolds number using the radius of the cylinder as a length scale rather than the average film thickness as was done by Deibler and Cerro. Thus, the dimensionless groups used by Orr and Scriven are the Reynolds number, $Re_{Orr} \equiv \rho\omega R^2/\mu$; the Weber number, $We_{Orr} \equiv \sigma/\rho\omega^2 R^3$; the inverse of the Froude number, $G_{Orr} = 1/Fr \equiv g/\omega^2 R$; and the ratio of the surface radius to the cylinder radius, $f_o = 1-\delta = (R-b)/R$.

Although the Reynolds number and Froude number ranges studied by Orr and Scriven are similar, when expressed in the same dimensionless parameters, the film thickness ratios studied by Orr and Scriven are much larger. Orr and Scriven looked at film thickness ratios of $\delta = 0.5$ and $\delta = 0.2$ while we are interested in thickness ratios closer to $\delta = 0.01$.

We have studied flow inside a cylinder with a range of Reynolds numbers from 100 to 3000, and a range of Froude numbers from 14 to 85. The Capillary number was not directly controlled as a parameter but was allowed to vary with the angular velocity of the cylinder (the physical properties and R remained constant throughout). For a typical dryer cylinder radius of 0.75 m, the range of Froude numbers studied correspond to paper machine speeds from 400 to 1500 m/min.

The initial condition used was to evenly distribute the fluid as a film along the edge of the cylinder rotation with the constant angular velocity. This initial condition

corresponds to a "solid body" rotation and is equivalent to a case with zero gravity. In early cases, the gravitational field was impulsively enforced at t = 0, which led to large oscillations in the surface position resulting in long simulation times for these oscillations to damp-out and steady-state to be reached. Later simulations have been conducted using a modified gravitational force

$$g(t) = \bar{g}\min(1, \alpha t) \tag{115}$$

where $\bar{g}$ is the steady state acceleration due to gravity, and α is a proportionality factor controlling the rate of change in the gravitational field. This approach is similar to that used by Orr and Scriven[109] to achieve convergence in their Finite Element simulations. The modifications needed for the gravitational force are accomplished through the use of the *tilde1.pat* and *tilde2.pat* shown in Tables 22 and 23, respectively. The surface profile is stored using an addition to draw.pat similar to those used in the die-swell problem and the coating problems discussed above.

Table 22. Modification to *tilde1.pat* for variable gravitational force in dryer cylinder.

```
temp1=gxa(j)
gxa(j)=temp1*dmin1(1.0d0,t*0.5d0)
temp2=gya(j)
gya(j)=temp2*dmin1(1.0d0,t*0.5d0)
```

Table 23. Modification to *tilde2.pat* for variable gravitational force in dryer cylinder.

```
gxa(j)=temp1
gya(j)=temp2
```

The boundary conditions are no-slip along the outer wall of the cylinder, periodic conditions between 0 and 2π in the θ-direction, Laplace's formula at the interface, and symmetry conditions on the front and back walls.

Determination of Grid Independence

The case at RE = 3000 and Fr = 85 was used to test the dependence of the solution on the size of the computational grid. Initially, 25 computational cells were used in the r-direction and 100 equally spaced computational cells were used in the θ-direction as indicated in the input data presented in Table 24. In order to obtain adequate resolution of the viscous sublayer, the minimum cell spacing in the r-direction adjacent to the wall of the cylinder was maintained as 1% of the average film thickness. A similar case was studied with 50 and 200 computational cells in the r and θ-directions, respectively.

Table 24. Input data for condensate flow problem with Re = 3000 and Fr = 85.

```
$xput
name=' Condensate flow in a rotating cylinder. Re=3000 Fr=85',
jnm=' test    ', nfcal=1, iequib=0, idefm=1, cyl=1.0d0,
velmx=1.0d0, isor=0, epsi=1.0d-3, lpr=1, cangle=90.0d0,
isurft=1, istress=1, alpha=0.5d0, flht=0.0d0,
omg=1.0d0, autot=0, icsurf=1, vi=25.0077488d0, gy=-9.81,
prtdt=1.0d0, tlimd=0.0d0, td=-1, t=0.0d0, dtcrmx=0.0005d0,
wf=4, wbk=4, wr=2, wl=2,
rhof=958.4d0, sigma=0.0000626d0, nu=0.000000294d0,
vrw=25.0077488d0,
delt=0.0005d0, pltdt=0.0471093d0, twfin=18.8437153d0, tddt=9.421858d0
$end
$meshgn
nkx=1,xl=0.739738d0, 0.75d0, xc=0.0.749948569d0, nxl=24, nxr=1,
dxmn=.000051431d0,
nky=1,yl=0.0d0, 4.7124d0, yc=2.3562d0, nyl=50, nyr=50, dymn=1.0d0,
nkz=1,zl=0.0d0, 0.01d0,   zc=0.005d0,  nzl=1,  nzr=1,  dzmn=1.0d0,
$end
$fluidgn
nqbs=1, qa1=-1.0d0, qa2=0.0d0, qb1=0.0d0,
qb2=0.0d0, qc1=0.7448569d0, qc2=0.0d0, iqh=1,
$end
```

Figure 30 shows a comparison of the results from the coarse and fine grid simulations in the viscous sublayer region adjacent to the cylinder wall at Re = 3000, Fr = 85. The ordinate and abscissa are non-dimensional quantities defined as a

dimensionless film thickness,

$$\eta = (R - r)\sqrt{\omega/2\nu}, \tag{116}$$

and the velocity deviation from the solid body rotation scaled by the Froude number,

$$\xi = (v_\theta/(R\omega) - 1)Fr, \tag{117}$$

respectively. The coarse grid results are in extremely good agreement with the results obtained using the fine grid, indicating that the coarse grid contains sufficient resolution for this problem. Therefore, computational grids with 25 by 100 computational cells were used in all subsequent simulations.

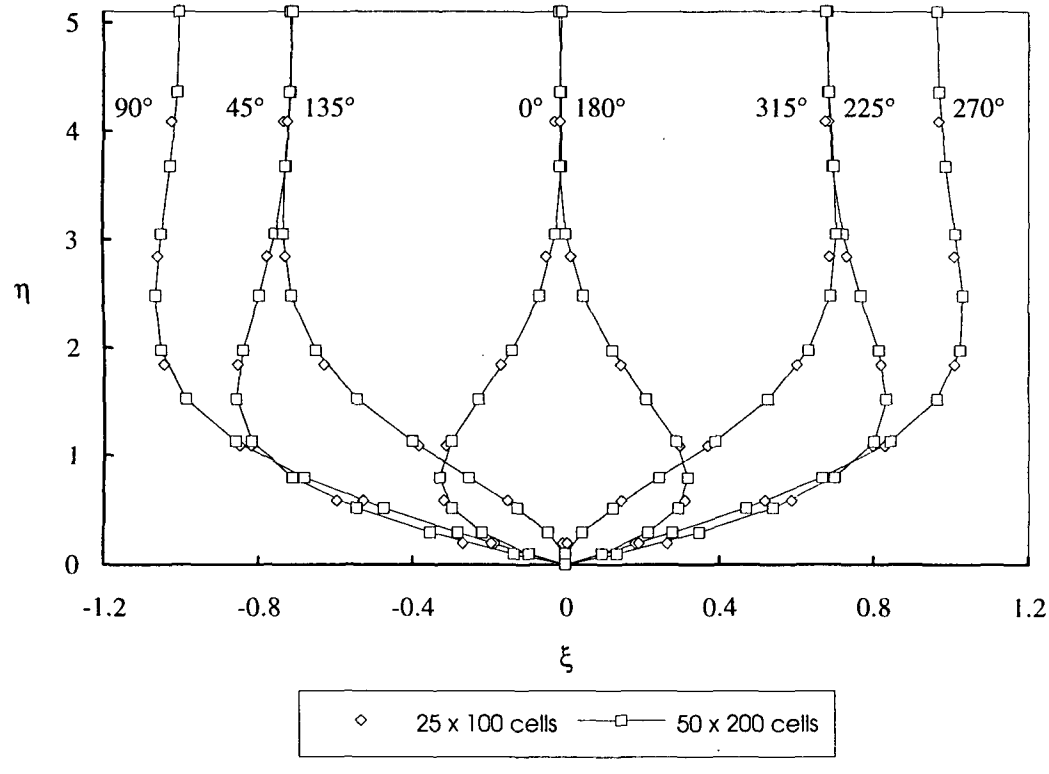

Figure 30. Comparison velocity profiles in the viscous sublayer at Re = 3000 and Fr = 85 results on different computation grids.

The velocity profiles shown in Figure 30 present only the viscous sublayer. The complete profiles from the wall to the interface are presented in Figure 31. Notice that the complete flow is made up of the viscous sublayer and a fairly large constant velocity region termed the inviscid core.

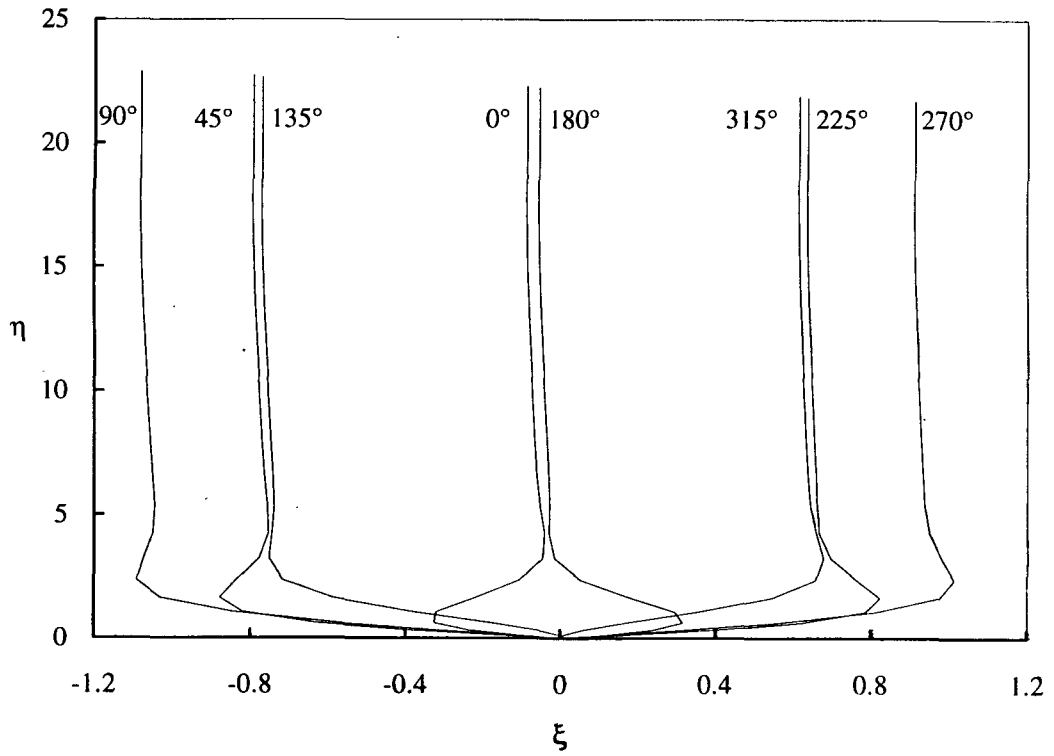

Figure 31. Complete velocity profiles at Re = 3000 and Fr = 85 (25 x 100 cells).

The Effect of Froude Number

Once a computational grid with sufficient resolution for accurate solution of the flow patterns was determined, a variety of cases were studied. First, I present a comparison of surface profile deviations from the solid body rotation for Re = 3000 as a function of Froude number (Figure 32). As the Froude number (the ratio of centrifugal

and gravitational forces) decreases, the free surface profile departs more and more from the solid body rotation, which is equivalent to Fr = ∞.

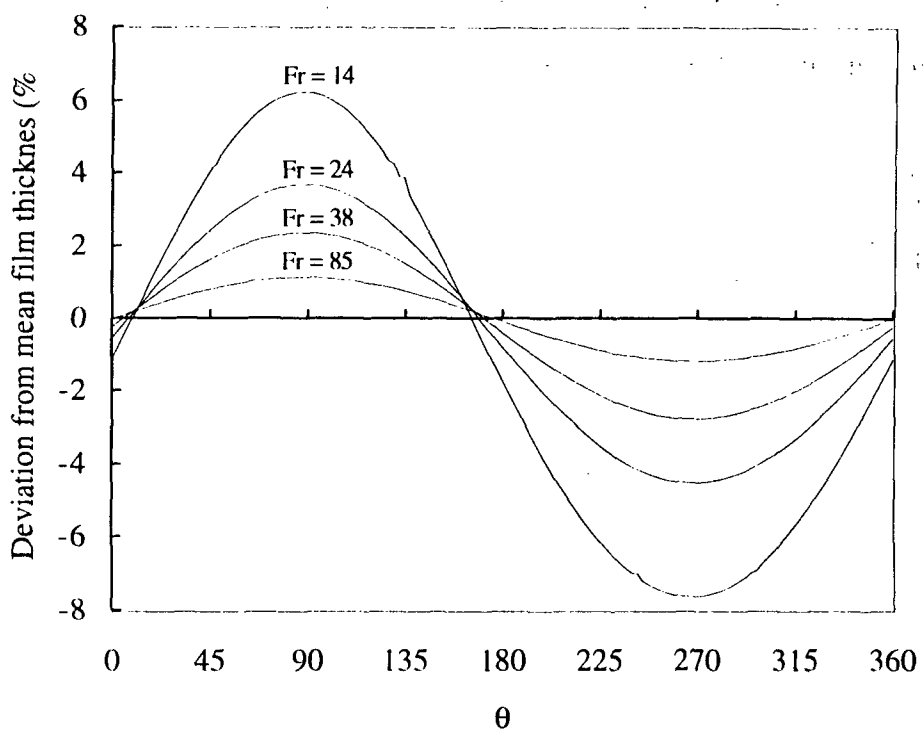

Figure 32. The effect of Froude number on condensate film thickness variation.

The remaining question is the effect of decreasing Froude number on the velocity profiles. For Fr = ∞, the velocity profiles are flat and constant as a function of position. As is shown in Figure 30 above, even at Fr = 85 there is significant deviation from the solid body solution. In Figure 33 we present velocity profiles for cases with Re = 3000 and Froude numbers of 85, 38, 24, and 13 at 0°, 45°, 90° and 135°.

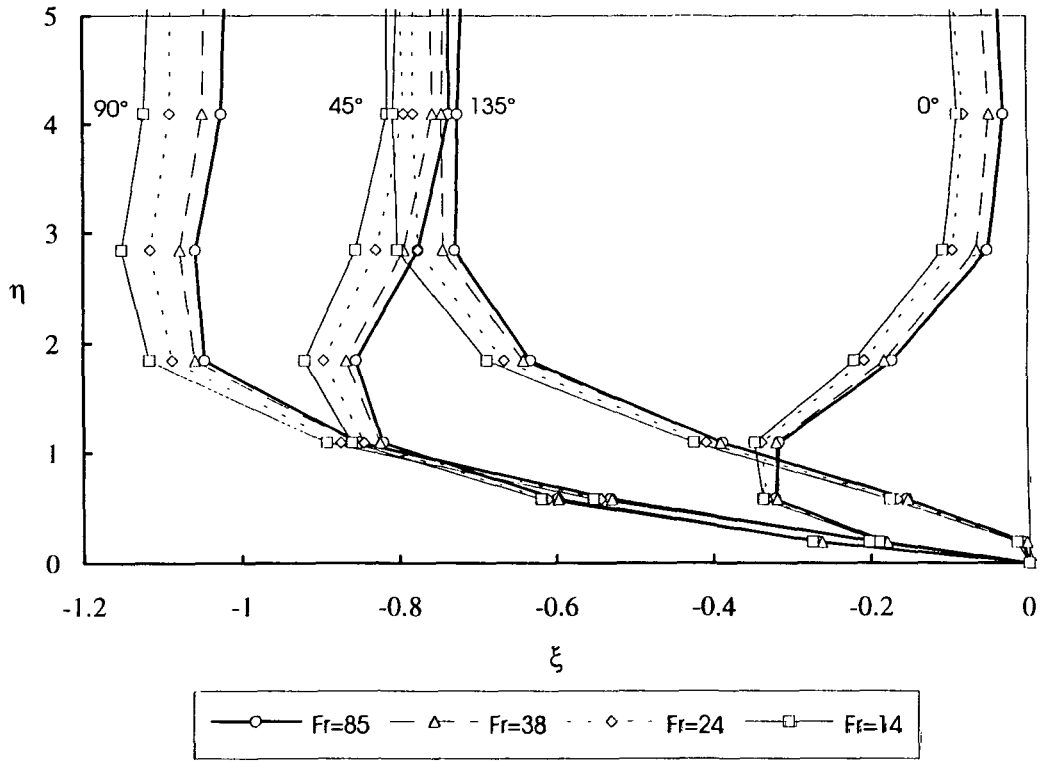

Figure 33. Comparison of velocity profiles for Re = 3000 at various Froude numbers.

The Effect of Reynolds Number

A similar comparison to that made in Figure 33 is possible to study the effect of Reynolds number at a specific Froude number. Figure 34 shows the effect of varying Reynolds number on velocity profiles at a constant Froude number. At a given Froude number the only parameter that was varied when the Reynolds number changes is the average film thickness. The results in Figure 34 show that the velocity profiles in the viscous sublayer remain constant as a function of Reynolds number at a given Froude number. Thus, increasing the Reynolds number (which is equivalent to increasing the average film thickness) does not affect the flow in the viscous sublayer, but only adds fluid to the inviscid core.

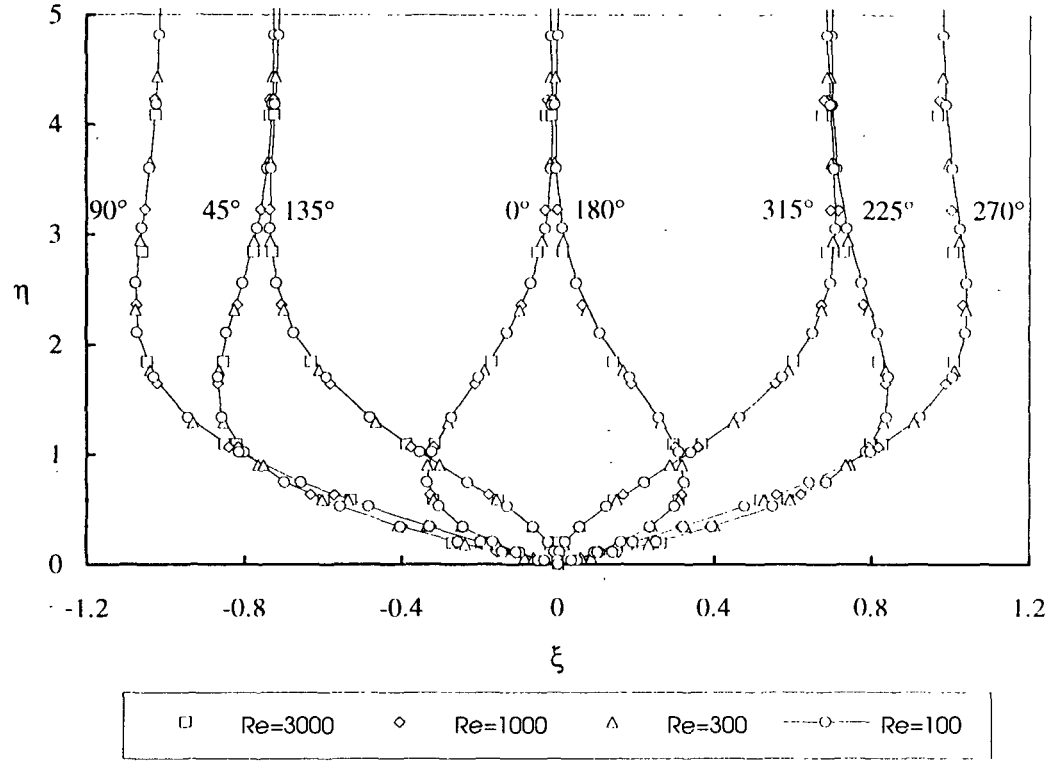

Figure 34. Comparison of velocity profiles for Fr = 85 at various Reynolds numbers.

Comparison with Experimental Results

As a further test of the accuracy of the computational technique for analysis of condensate flows is to compare the computational results with those from experiment. Limited experiments using an ultrasonic film thickness technique were performed at the University of Lund in Sweden.[110] Presented in Figures 35 and 36 are comparisons of the computed surface profiles with the experimentally measured film thickness for case at Re = 763, Fr = 6 and Re = 1107, Fr = 13, respectively. The experimental data are presented as a series of three points recorded every 45° around the cylinder. No error bars were available, so the reported maximum and minimum values represent the range of recorded thicknesses with the middle point representing the average reading. The trends

in the film thickness and the general magnitude of the deviations from the solid body rotation show good agreement between the computed and experimental results.

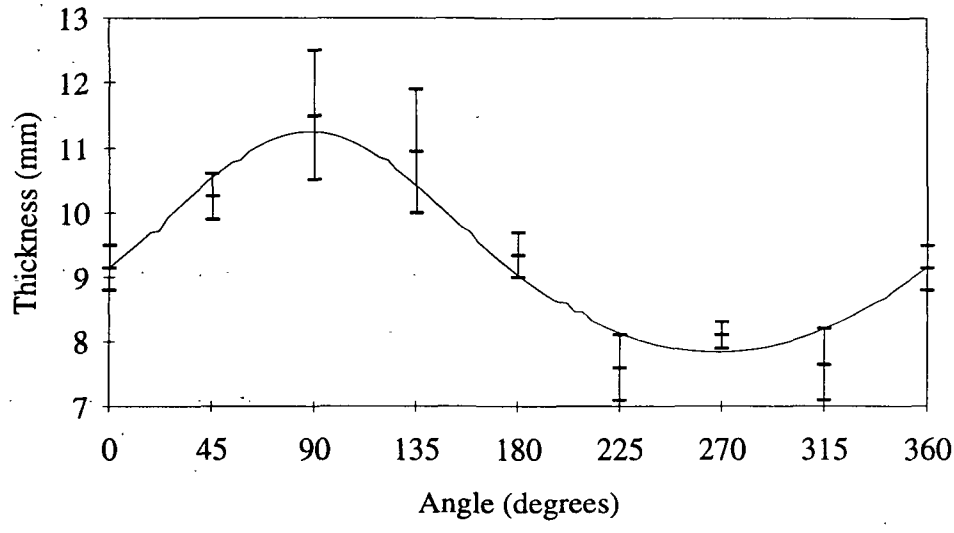

Figure 35. Comparison of computed and experimental[110] for Re = 763 and Fr = 6.

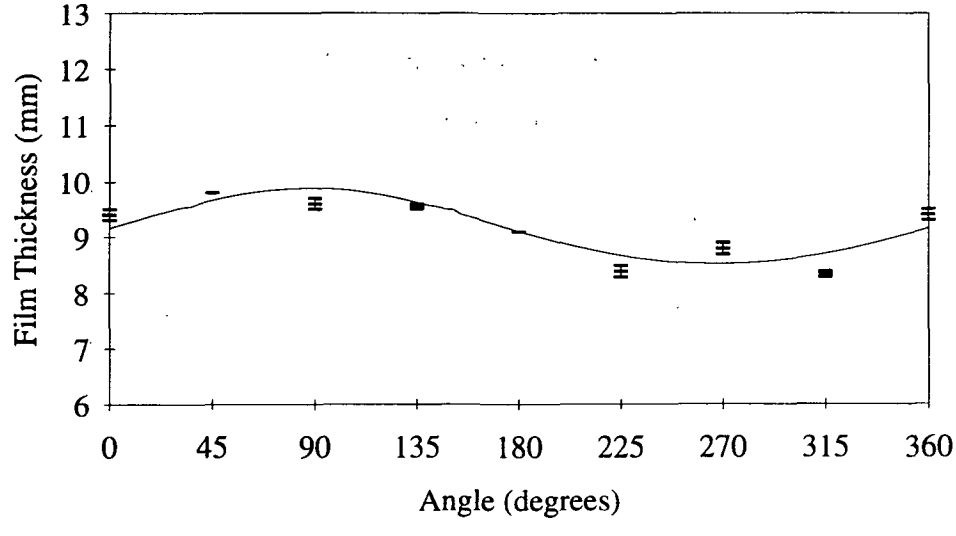

Figure 36. Comparison of computed and experimental[110] for Re = 1107 and Fr = 13.

Implications for Heat Transfer

The flow in the condensate layer is characterized as a relatively small viscous sublayer supporting a much larger inviscid core having a nearly constant velocity profile. From Figure 33 it is clear that as the machine speed, and thus the Froude number, is increased, the velocity profiles approach the solid body rotation. Accompanying this is a shift towards a constant film thickness seen in Figure 32.

Since there is no mixing within the condensate layer, the heat transfer mechanism through the condensate layer is expected to be by conduction. In general, convection is superior to conduction as a heat transfer mechanism, so some means of introducing mixing and thus increasing convection is expected to have a significant effect on the heat transfer rate through the condensate layer.

There are several ways to optimize the heat transfer through the condensate layer. Decreasing the thickness of the condensate layer to reduce the distance through which the heat must be conducted is one approach. Perhaps the most cost-effective method of increasing the heat transfer rate through the condensate layer is to promote mixing and thus convection within the condensate layer. An example of this is the use of axial bars on the inner surface of the dryer cylinder. Pulkowski and Wedel[111] present examples of the dramatic effect these "spoiler" bars can have on the heat transfer rate.

CONCLUSIONS

Free surface flows exist in a number of pulp and papermaking unit operations including the spraying of black liquor into a recovery furnace, jets leaving the headbox, the paper machine forming section, condensate flows inside dryer cylinders, coating application systems, and finishing operations such as polymer film extrusion. Specifically, this work has been motivated by the study of the stability of a thin viscous sheet flowing through a stagnant inviscid vapor phase. This problem is directly related to black liquor spraying, an important part of the operation of a recovery furnace.

In the course of this work, a computational fluid dynamics model for transient three-dimensional free surface flows has been developed with capabilities beyond previously existing computational techniques. This was accomplished by enhancing and extending the capabilities of the SOLA-VOF computational technique in the following manner:

- Improved accuracy of the advective terms in the NSE through the addition of three third-order accurate finite differencing schemes.
- The capability to treat static contact points (or lines) by treating both the wall adhesion force due to surface tension and velocity boundary conditions in the region of the static contact point (or line).
- Modification of the free surface boundary condition arising from the normal stress balance to include the liquid phase deviatoric normal stress at the interface.
- Added a computational technique to allow the pressure in the vapor phase to vary when the vapor phase is assumed to be inviscid and irrotational and thus governed by potential flow.

Each of these enhancements to the computation technique was tested and compared with results from the literature. First, the lid-driven cavity problem

demonstrated the accuracy of the third-order accurate differencing schemes for the advective terms in the NSE. Next, the die-swell problem tested the accuracy of the static contact line treatment and the inclusion of the liquid phase deviatoric normal stress in the interfacial boundary condition. Finally, study of the growth of waves in a thin viscous sheet of liquid flowing through a stagnant inviscid vapor phase showed the accuracy of the numerical technique for allowing the pressure in the vapor phase to vary.

Additional applications of the computational technique to problems of interest to the pulp and paper have been demonstrated. First, analysis of the effects of temporal and spatial pressure fluctuations upstream of a coating blade on the thickness of the coating layer was accomplished. Second, the effect of varying Froude number and Reynolds number on the flow of condensate in a paper dryer cylinder was analyzed. These applications demonstrate that the IPST-VOF3D computational technique in its current form can assist in understanding phenomena of practical interest to the pulp and paper industry.

With the development of IPST-VOF3D, a general computational technique is available to solve three-dimensional free surface problems involving complex geometries and potentially intersecting free surfaces. Example problems have been presented demonstrating the accuracy and capabilities of the computational technique. The IPST-VOF3D computational technique is unique in its ability to accurately solve the coupled problem of an initially stagnant, inviscid phase and moving viscous phase.

In summary, Table 25 compares the features of IPST-VOF3D with the features of two commercially available computational fluid dynamics programs having free surface capabilities, FIDAP[71] and NEKTON.[72]

Table 25. Features of the IPST-VOF3D, FIDAP, and NEKTON programs.

Feature	IPST-VOF3D	FIDAP 7.0	NEKTON 2.85
Interface tracking	VOF (can handle large interface deformation)	Method of Spines Local perturbation (new in 7.0)	Mapping
Two fluids	yes (if one fluid is governed by potential flow)	yes (limitations not known to author)	yes (if fluid densities are similar)
Turbulence	no	mixing length k-ε	mixing length k-ε
Three-dimensional	yes	yes (new in 7.0)	yes
Transient	yes	yes	yes
Energy equation	no	yes	yes
Phase change	no	liquid-solid	liquid-solid
Non-Newtonian	no	Bingham Fluid Power law Carreau Model	User Supplied
Variable surface tension	no	yes	yes

AREAS FOR ADDITIONAL WORK

In this section I discuss areas for additional work directly resulting from this thesis. Specifically, I outline additional features that should be added to the IPST-VOF3D computational technique to enhance its capabilities. These additions are primarily motivated by paper industry free surface flow problems that cannot be properly addressed with the current capabilities of the IPST-VOF3D computational technique.

Many paper industry problems consist of non-Newtonian flows, including coating flows and pulp suspensions. Thus, the addition of options for non-Newtonian fluids would be of great use. Several of the choices made in the computational technique development have been made with this in mind, e.g., the choice to use the full deviatoric stress at the interface rather than the simplified formula derived in Appendix V.

One of the most important free surface flows in the papermaking process is the turbulent jet issuing from the headbox. Therefore, an extremely important extension to the IPST-VOF3D computational technique is the addition of a turbulence modeling capability.

Currently, the computational technique assumes that all flows are isothermal and that the fluid properties are constant. More complete analysis of problems, such as the flow of condensate in a dryer cylinder, can be accomplished through the addition of the energy equation and physical properties that vary with temperature. This may also require modification of the boundary conditions at the interface between the liquid and vapor phases, since the assumption of zero tangential stress may no longer be valid when the surface tension is allowed to vary. A mechanism for treating phase change would also enhance the ability of this computational technique to address heat transfer problems.

An additional area in need of study is extension of the continuum surface force (CSF) model for treating surface tension.[13] As discussed above, this technique converts the discontinuity in pressure at the interface caused by the surface tension into an equivalent localized body force. This procedure allows more accurate treatment of the force due to surface tension, but has currently only been implemented for flows where the boundary condition can be governed by Laplace's formula, i.e., without the liquid phase deviatoric normal stress. Thus, even if the surface tension force is included using the CSF model, a discontinuity is still required due to the presence of the liquid phase deviatoric normal stress. It may be possible to formulate an analogy to the CSF model that would include the effects of the deviatoric stress as a localized body force.

The final addition to the code that is warranted is the extension of the vapor phase potential solution to cylindrical coordinates. This is necessary for study of the stability of radially thinning viscous liquid sheets flowing through a stagnant vapor phase, a problem directly applicable to the process of black liquor spraying.

ACKNOWLEDGMENTS

I would like to thank the Institute of Paper Science and Technology, its member companies, and alumni for their support throughout this work. The computations were conducted, in part, using the Cornell National Supercomputer Center, a resource of the Center for Theory and Simulation in Science and Engineering at Cornell University, which is funded in part by the National Science Foundation, New York State, and the IBM Corporation.

I would like to dedicate this work to my friend and wife, Teri Ann McKibben, for all of her love, patience, and support, without which this work would not have been possible; to my son, Sean Patrick McKibben, who always has a smile for me no matter how bad my day is; and to my parents, Captain Ferney Marvin McKibben USCG Retired and Mary Leach McKibben, for teaching me the value of a good education and giving me the desire to become an engineer.

I would like to thank Hiroshi Miura from Mitsubishi Heavy Industries for his assistance in the analysis of coating flows, Björn Wilhelmsson and Dr. Stig Stenström from the University of Lund for working with me on the condensate flow problem and providing the experimental data for comparison, and Dr. Agnes Kovacs for her diligence in reading my dissertation and for continuing to develop this computational technique.

Finally, I would like to thank all of the faculty members who have served on my committee, especially Dr. Jeff Lindsay, for his support in this work from the beginning, and my advisor Dr. Cyrus Aidun for helping me to be rigorous in my approach and instilling in me higher expectations for my work.

NOMENCLATURE

VECTORS AND TENSORS

g	body force vector (includes gravitational forces)
i	x-direction unit vector in Cartesian coordinates
j	x-direction unit vector in Cartesian coordinates
k	x-direction unit vector in Cartesian coordinates
n	unit vector normal to the interface
r	r-direction unit vector in cylindrical coordinates
s	unit vector tangent to the interface and tangent to the **t**
t	unit vector tangent to the interface
u	velocity vector (in Cartesian or cylindrical coordinates)
$\tilde{\mathbf{u}}$	explicit guess for velocity vector
z	z-direction unit vector in cylindrical coordinates
θ	θ-direction unit vector in cylindrical coordinates
τ	deviatoric stress tensor (viscous component of the total stress tensor)

SCALARS

a	initial sheet half-thickness in wave growth problem
AR	aspect ratio: defined as needed
b	average film thickness
Ca	capillary number: defined as needed
D	divergence: $D = \nabla \cdot \mathbf{v}$
F	VOF function, fraction of a computational cell containing fluid
Fr	Froude number
g	acceleration due to gravity
h	dimension of problem: defined as needed
H	interfacial position function
i	imaginary number $i = \sqrt{-1}$
I	x or r-direction index
J	y or θ-direction index
k	wavenumber of disturbance
K	z-direction index
m	dimensionless wavenumber: $m = ka$
p	pressure
P	reduced pressure: $P = p/\rho$
r	x-direction component in cylindrical coordinates
or	scale factor: r=1 in Cartesian coordinates and r=x in cylindrical coordinates

R		r-direction interfacial position function
	or	radius of a cylinder
Re		Reynolds number: defined as needed
t		time
u		velocity component in the x or r-direction
U_0		initial sheet velocity
v		velocity component in the y or θ-direction
V		characteristic velocity: defined as needed
w		velocity component in the z-direction
W		width of lid driven cavity
	or	characteristic velocity: defined as needed
We		Weber number: $We_\ell = \rho_\ell U_0^2 a/\sigma$
x		x or r coordinate direction or position
y		y or θ coordinate direction or position
z		z coordinate direction or position
Z		z-direction interfacial position function
	or	Ohnesorge number: $Z = \mu_\ell (\rho_\ell a \sigma)^{-1/2}$
α		fraction of upwind differencing
α′		modified upwind differencing fraction: $\alpha' = \alpha$ sign(local velocity)
β		SOR factor
δt		time step
Δx		local computational cell width
Δy		local computational cell depth
Δz		local computational cell height
ε		perturbed interface position
η		local height function
	or	interpolation factor
	or	dimensionless position
κ		total curvature of the interface between two fluids
ν		kinematic viscosity
μ		viscosity
θ		y-direction component in cylindrical coordinates
Θ		θ-direction interfacial position function
	or	partial cell function
ρ		density
$\tilde{\rho}$		Density ratio: $\tilde{\rho} = \rho_v/\rho_\ell$
σ		surface tension between two fluids
ω		wave growth rate
	or	SOR acceleration factor
	or	angular rotation of a cylinder

$\tilde{\omega}$ dimensionless growth rate: $\tilde{\omega} = \tilde{\omega}_r + i\,We_\ell^{1/2}\tilde{\omega}_i$

$\tilde{\omega}_1$ modified dimensionless growth rate: $\tilde{\omega}_1 = \tilde{\omega} + i\,We_\ell^{1/2}m$

$\tilde{\omega}_r$ real part of dimensionless growth rate: $\tilde{\omega}_r = \omega_r\left(\sigma/\rho_\ell a^3\right)^{-1/2}$

$\tilde{\omega}_i$ imaginary part of dimensionless growth rate: $\tilde{\omega}_i = \omega_i\left(a/U_0\right)m$

ξ dimensionless velocity

ζ scale factor: ζ=0 in Cartesian coordinates and ζ=1 in cylindrical coordinates

SUBSCRIPTS

0 constant or base condition

i imaginary part of growth rate
or x or r-direction index

j y or θ-direction index

k z-direction index

ℓ liquid phase

L left side of computational domain (smallest position in x-direction)

n neighboring cell

r derivative with respect to the r-direction in cylindrical coordinates
or real part of growth rate

R right side of computational domain (largest position in x-direction)

s surface

v vapor phase

x derivative with respect to the x-direction in Cartesian coordinates

y derivative with respect to the y-direction in Cartesian coordinates

z derivative with respect to the z-direction in Cartesian or cylindrical coordinates

θ derivative with respect to the θ-direction in cylindrical coordinates

SUPERSCRIPTS

n time step

x height function normal to the x-direction

y height function normal to the y-direction

z height function normal to the z-direction

ν iteration

LITERATURE CITED

1. Hough, G. ed. Chemical Recovery in the Alkaline Pulping Processes. TAPPI Press, Atlanta, 1985.

2. Adams, T. N. and Frederic, W. J. Kraft Recovery Boiler Physical and Chemical Processes. The American Paper Institute, Inc., New York, 1988.

3. Frederick, W. J. Combustion Processes in Black Liquor Recovery: Analysis and Interpretation of Combustion Rate Data and Engineering Design Model. Report No. One, Department of Energy Contract No. AC02-83CE40637, (March 1990).

4. Miller, P. T. Swelling of Kraft Black Liquor. Doctoral Dissertation. The Institute of Paper Chemistry, Appleton, WI, 1986.

5. Kulas, K. A. An Overall Model of the Combustion of a Single Droplet of Kraft Black Liquor. Doctoral Dissertation. The Institute of Paper Chemistry, Appleton, WI, 1990.

6. Spielbauer, T. M. The Stability and Disintegration of Radially Thinning Liquid Sheets. Doctoral Dissertation. Institute of Paper Science and Technology, Atlanta, GA, 1992.

7. Hirt, C. W.; Nichols, B. D.; Romero, N. C. SOLA—A Numerical Solution Algorithm for Transient Fluid Flows. Los Alamos National Laboratory Report LA-5852, 1975.

8. Hirt, C. W.; Nichols, B. D. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. J. of Comput. Physics, 39:201-25(1981).

9. Navier, M. Mem. del'Acad. des Sciences, 6:389(1822).

10. Stokes, G. G. Trans. Cambridge Phil. Soc., 8:287(1845).

11. Batchelor, G.K. An Introduction to Fluid Dynamics. Cambridge University Press, (1967).

12. Aidun, C. K. Mechanics of a Free-Surface Liquid Film Flow, J. Appl. Mech. 54:951-954(1987).

13. Brackbill, J. U., Kothe, D. B., and Zemach, C. A Continuum Method for Modeling Surface Tension. J. Comput. Phys. 100:335-354(1992).

14. Floryan, J. M. and Rasmussen, H. Numerical Methods for Viscous Flows with Moving Boundaries. Appl. Mech. Rev. 42(12):323-41(1989).

15. Floryan, J. M. and Rasmussen, H. A Discussion of Numerical Methods for Viscous Free Surface Flows. Free Boundary Problems: Theory and Applications, Vol. II. Longman Scientific and Technical, 778-787, 1990.

16. Yeung, R. W. Numerical Methods in Free Surface Flows. Ann. Rev. Fluid Mech., 14:395-442(1982).

17. Hyman, J. M. Numerical Methods for Tracking Interfaces. Physica, 12D:369-407(1984).

18. Laskey, K. J., Oran, E. S., and Boris, J. P. Approaches to Resolving and Tracking Interfaces and Discontinuities. Naval Research Laboratory Report MR-5999, Arlington, VA 1987.

19. Crank, J. Free and Moving Boundary Problems, Clarendon, Oxford, 1984.

20. Bulgarelli, U., Casulli, V., and Greenspan, D. Pressure Methods for the Numerical Solution of Free Surface Flows, Pineridge, Swansea, UK, 1984.

21. Harlow, F. Numerical Methods for Fluid Dynamics, An Annotated Bibliography, Los Alamos National Laboratory Report LA-4281, 1969.

22. Tseng, A. A., Zou, J., Wang, H. P., and Hoole, S. R. H. Numerical Methods of Macro and Micro Behaviors of Materials in Processing: A Review. J. Comput. Phys. 102:1-17(1992).

23. Hirt, C. W., Cook, J. L., and Butler, T. D. A Lagrangian Method for Calculating the Dynamics of an Incompressible Fluid with Free Surface. J. Comput. Phys. 5:103-124(1970).

24. Butler, T. D. LINC method extensions, in Lecture Notes in Physics. Springer-Verlag, New York, 8:435-440(1971).

25. Crowley, W. P. FLAG: A Free-Lagrangian Method for Numerically Simulating Hydrodynamic Flows in Two-Dimensions, in Lecture Notes in Physics. Springer-Verlag, New York, 8:37-43(1971)

26. Free-Lagrangian Methods for Compressible Hydrodynamics in Two Space Dimensions, in Lecture Notes in Physics. Springer-Verlag, New York, 238:1-21(1985)

27. Fritts, M. J. and Boris, J. P. The Lagrangian Solution of the Transient Problems in Hydrodynamics using a Triangular Mesh, J. Comput. Phys. 31:173-215(1979).

28. Fritts, M. J., Crowley, W. P., and Trease, H. E. Eds. The Free-Lagrange Method. Lecture Notes in Physics, Springer-Verlag, New York, 238:(1985).

29. Trease, H. E. Three-Dimensional Free-Lagrangian Hydrodynamics, in Lecture Notes in Physics. Springer-Verlag, New York, 238:145-157(1985).

30. Fyfe, D. E., Oran, E. S., and Fritts, M. J. Numerical Simulation of Droplet Oscillations, Breakup, and Distortion. AIAA Paper AIAA-87-0539, 1987.

31. Fyfe, D. E., Oran, E. S., and Fritts, M. J. Surface Tension and Viscosity with Lagrangian Hydrodynamics on a Trianguar Mesh. J. Comput. Phys., 76:349-384(1987).

32. Clark, R. A. Compressible Lagrangian Hydrodynamics without Lagrangian Cells, in Lecture Notes in Physics. Springer-Verlag, New York, 238:145-157(1985).

33. Clark, R. A. Free-Lagrangian Hydrodynamics using Massless Tracer Points, in Lecture Notes in Physics. Springer-Verlag, New York, 264:181-185(1986).

34. Hirt, C. W., Amsden,. A. A., and Cook, J. L. An Arbitrary Lagrangian Eulerian Computing Method for all Speeds. J. Comput. Phys. 14:227-253(1974).

35. Amsden, A. A., Ruppel, H. M., and Hirt, C. W. SALE: A Simplified ALE Computer Program for Fluid Flow at All Speeds. Los Alamos National Laboratory Report, LA-8095, 1980.

36. Addessio, F. L., Carroll, D. E., Dukowicz, J. K., Harlow, F. H., Johnson, J. N., Kashiwa, B. A., Maltrud, M. E., and Ruppel, H. M. CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip. Los Alamos National Laboratory Report, LA-10613-MS, 1986.

37. Bach, P. and Hassager, D. An Algorithm for the Use of the Lagrangian Specification in Newtonian Fluid Mechanics and Application to Free-Surface Flow. J. Fluid Mech. 152:173-190(1985).

38. Nichols, B. D.; Hirt, C. W. Calculating Three-Dimensional Free Surface Flows in the Vicinity of Submerged and Exposed Structures. J. of Comput. Phys. 12:234-246(1973).

39. Nichols, B. D.; Hirt, C. W. Improved Free Surface Boundary Conditions for Numerical Incompressible Flow Calculations. J. of Comput. Phys. 8:434-448(1971).

40. Hill, G. A. Numerical Simulation of Free Surface Flows. Doctoral Dissertation, University of Saskatchewan, Saskatoon, 1979.

41. Hill, G. A., Shook, C. A., and Esmail, M. N. Finite Difference Simulation of Die Swell for a Newtonian Fluid. Can. J. Chem. Eng. 59:100-108(1981).

42. Harlow, F. H.; Welch, J. E. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free-Surface. Phys. Fluids, 8(12):2182-9(1965).

43. Nichols, B. D. and Hirt, C. W. Methods for Calculating Multi-Dimensional, Transient, Free Surface Flows Past Bodies. Proceedings First International Conference on Numerical Ship Hydrodynamics. Kings Point, NY, 1975.

44. Amsden, A. A.; Harlow, F. H. A Simplified MAC Technique for Incompressible Fluid Flow Calculations. J. of Comput. Physics, 6:322-5(1970).

45. Viecelli, J. A. A Computing Method for Incompressible Flows Bounded by Moving Walls. J. of Comput. Physics, 8:119-43(1971).

46. Noh, W. F. and Woodward, P. SLIC (Simple Line Interface Calculation), in Lecture Notes in Physics. Springer-Verlag, New York, 59:330-340(1976).

47. Chorin, A. J. Flame Advection and Propagation Algorithms. J. Comput. Phys. 35:1-11(1980).

48. Chorin, A. J. Curvature and Solidification. J. Comput. Phys. 58:472-490(1985).

49. Nichols, B. D.; Hirt, C. W.; Hotchkiss, R.S. SOLA-VOF: A Solution Algorithm for Transient Fluid Flow with Multiple Free-Boundaries. Los Alamos National Laboratory Report LA-8355, 1980.

50. Lewis, R. W; Morgan, K.; Pugh, E. D. L.; Smith, T. J.; A Note on Discontinuous Numerical Solutions of the Kinematic Wave Equation. Int. J. Num. Meth. Eng., 20:555-63(1984).

51. Thompson, E.; Use of Pseudo-Concentrations to Follow Creeping Viscous Flows During Transient Analysis. Int. J. Num. Meth. Fluids, 6:749-61(1986).

52. Smith, T. J.; Welbourn, D. B.; The Integration of Geometric Modeling with Finite Element Analysis for the Computer-Aided Design of Castings. Appl. Sci. Res., 44:139-60(1987).

53. Usmani, A. S.; Cross, J. T.; Lewis, R. W.; A Finite Element Model for the Simulations of Mold Filling in Metal Casting and the Associated Heat Transfer. Inter. J. Num. Meth. Eng., 35:787-806(1992).

54. Kothe, D. B., Mjolsness, R. C., and Torrey, M. D. RIPPLE: A Computer Program for Incompressible Flows with Free Surfaces. Los Alamos National Laboratory Report, LA-12007-MS, 1991.

55. Kothe, D. B. and Mjolsness, R. C. RIPPLE: A New Model for Incompressible Flows with Free Surfaces. AIAA J. 30(11):2694-2700(1992).

56. Unverdi, S. O. and Tryggvason, G. A Front Tracking Method for Viscous, Incompressible Multi-Fluid Flows. J. Comput. Phys. 100:25-37(1992).

57. Unverdi, S. O. and Tryggvason, G. Computations of Multi-fluid Flows. Physica D. 60:70-83(1992).

58. Li, Xiangou and Tankin, R. S. On the Temporal Instability of a Two-Dimensional Viscous Liquid Sheet. J. Fluid Mech. 226:524-433(1991).

59. Tanner, R. I.; Extrudate Swell, in Computational Analysis of Polymer Processing, J. R. A. Pearson and S. M. Richardson, Eds, Applied Science, London, 1983.

60. Denn, M. M.; Fibre Spinning, in Computational Analysis of Polymer Processing, J. R. A. Pearson and S. M. Richardson, Eds, Applied Science, London, 1983.

61. Kistler, S. F.; Scriven, L. E.; Coating Flows, in Computational Analysis of Polymer Processing, J. R. A. Pearson and S. M. Richardson, Eds, Applied Science, London, 1983.

62. Kheshgi, H. S.; Scriven, L. E.; Penalty Finite Element Analysis of Unsteady Free Surface Flows. Finite Elements in Fluids, R. H. Gallagher et al., Wiley, New York, 5:393-434(1984).

63. Kroeger, P. G.; Ostrach, S.; The Solution of Two-Dimensional Freezing Problem Including Convection in the Liquid Region. Int. J. of Heat and Mass Transfer, 17:1191-207(1974).

64. Ryskin, G; Leal, L. G.; Numerical Solution of Free-Boundary Problems in Fluid Mechanics. Part 1. The Finite Difference Technique. J. of Fluid Mech., 148:1-17(1984).

65. Dutta, A.; Ryan, M. E.; Dynamics of a Creeping Newtonian Jet with Gravity and Surface Tension: A Finite Difference Technique for Solving Steady Free-Surface Flows Using Orthogonal Curvilinear Coordinates. AIChE J., 28(2):220-32(1982).

66. Nickell, R. E., Tanner, R. I., and Caswell, B. The Solution of Viscous Incompressible Jet and Free-Surface Flows using Finite Element Methods. J. Fluid Mech. 65:434-448(1974).

67. Reddy, K. R., and Tanner, R. I. Finite Element Solution of Viscous Jet Flows with Surface Tension. Comput. Fluids 6:83-91(1978).

68. Omodei, B. J. Computer Solutions of a Plane Newtonian Jet with Surface Tension. Comput. Fluids 7:79-96(1979).

69. Sillman, W. J.; Scriven, L. E.; Separating Flow near a Static Contact Line: Slip at a Wall and Shape of a Free-Surface. J. Comput. Phys., 34:287-313(1980).

70. Crochet, M. J. and Keunings, R. On Numerical Die Swell Calculation, J. Non-Newtonian Fluid Mech. 10:85-94(1982).

71. FIDAP User's Manual, Revision 7.0, Fluid Dynamics International, Inc. Naperville, IL (1993).

72. Nekton User's Manual, Version 2.8, Nektonics Inc. Cambridge MA (1991).

73. Harlow, F. The Particle-in-Cell Computing Method for Fluid Dynamics, in Methods in Computational Physics. Academic, New York, 3:117-179(1964).

74. Noh, W. F. CEL: A Time Dependent, Two-Space-Dimensional, Coupled Eulerian-Lagrangian Code. in Methods in Computational Physics. Academic, New York, 3:117-179(1964).

75. Dussan, E. B. The Moving Contact Line: The Slip Boundary Condition. J. Fluid Mech. 77:665-684(1976).

76. Hocking, L. M. A Moving Fluid Interface. Part 2. The Removal of the Force Singularity by a Slip Flow. J. Fluid Mech. 79:209-229(1977).

77. Huh, C. and Mason, S. G. The Steady Movement of a Liquid Meniscus in a Capillary Tube. J. Fluid Mech. 81:401-419(1977).

78. Greenspan, H. P. On the Motion of a Small Viscous Droplet that Wets a Surface. J. Fluid Mech. 84:125-143(1978).

79. Huh, C. and Scriven, L. E. Hydrodynamic Model of Steady Movement of a Solid/Liquid Fluid Contact Line. J. Colloid. Interface Sci. 35:85-101(1971).

80. Torrey, M. D.; Cloutman, L. D.; Mjolsness, R. C; Hirt, C. W. NASA-VOF2D: A Computer Program for Incompressible Flows with Free Surfaces. Los Alamos National Laboratory Report LA-10612-MS, 1985.

81. Torrey, M. D.; Mjolsness, R. C.; Stein, L. R. NASA-VOF3D: A Three-Dimensional Computer Program for Incompressible Flows with Free Surfaces. Los Alamos National Laboratory Report LA-11009-MS, 1987.

82. Squire, H. B. Investigation of the Instability of a Moving Liquid Film. Brit J. of Appl. Physics, 4:167-169(1953).

83. Hagerty, W. W. and Shea, J. F. A Study of the Stability of Plane Fluid Sheets. J. Appl. Mech., 22:509-514(1955).

84. Roach, P. J. Computational Fluid Dynamics. Hermosa Publishers, Albuquerque, NM, 1976.

85. Leonard, B. P.; A Stable and Accurate Convective Modeling Procedure Based on Quadratic Upstream Interpolation. Computer Methods in Applied Mechanics and Engineering, 19:59-98(1979).

86. Agarwal, R. K.; A Third-Order-Accurate Upwind Scheme for Navier-Stokes Solutions at High Reynolds Numbers, AIAA Paper AIAA-81-0112, 1981.

87. Kawamura, T. and Kuwahara, K.; Computation of High Reynolds Number Flow around a Circular Cylinder with Surface Roughness, AIAA Paper AIAA-84-0340, 1984.

88. Peyret, R and Taylor, T. D. Computational Methods for Fluid Flow. Springer Series in Computational Physics, Springer-Verlag, New York, 1983.

89. Fletcher, C. A. J. Computational Techniques for Fluid Dynamics: Volume II. Springer Series in Computational Physics, Springer-Verlag, New York, 1988.

90. Chandra, R. Conjugate Gradient Methods for Partial Differential Equations. Doctoral Dissertation, Yale University, 1978.

91. Panton, R. L. Incompressible Flow. Wiley, New York, 1984.

92. Kantorovich, L. and Krylov, V. Approximate methods of Higher Analysis. Noordhoff Ltd., Netherlands, 1958.

93. Bramble, J. H. and Hubbard, B. E. Approximation of Solutions of Mixed Boundary Value Problems for Poisson's Equation by Finite Differences. J. Assoc. Comput. Mach. 12:114-123(1965).

94. VanLinde, H. J., High-Order Finite-Difference Methods for Poisson's Equation. Doctoral Dissertation, University of Groningen, Netherlands, 1971.

95. VanLinde, H. J., High-Order Finite-Difference Methods for Poisson's Equation. Math. of Comput. 28(126):369-391(1974).

96. The Data Visualizer™ Programming Guide, Version 3.0, Wavefront Technologies Inc., Santa Barbara, CA, 1993.

97. Miura, H. and Aidun, C. K. Pressure Fluctuations and Coat-Weight Nonuniformities in Blade Coating. 1992 TAPPI Coating Conference Proceedings, TAPPI Press, pp. 193-216, 1992.

98. Miura, H., McKibben, J. F., and Aidun, C. K. Effects of CD Pressure Variations on Coat-Weight Non-Uniformities. (Under Preparation).

99. Wilhelmsson, B. I., McKibben, J. F., Stenström, S. G., and Aidun, C. K. Condensate Flow Inside Paper Dryer Cylinders. (Under Preparation).

100. Ghia, U.; Ghia, K. N.; Shin, C. T.; High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Solver. J. of Comput. Phys., 48:387-411(1982).

101. Kawai, M. and Ando, Y. Comparison of Third-Order Upwind Schemes at High Reynolds Numbers. Nippon Kakai Gakkai Ronbunshu, B. Hen, 52(477):2067-2071(1985).

102. Dupret, F. A Method for the Computation of Viscous Flow by Finite Elements with Free Boundaries and Surface Tension, in Finite Element Flow Analysis, T. Kawai Ed. University of Tokyo Press, Tokyo, 1982.

103. Triantafillopoulos, N. G. and Aidun, C. K. Relation Between Flow Instability in Short-Dwell Ponds and Cross Directional Coat Weight Nonuniformities. TAPPI J. 73,127(1990).

104. Li, A. Personal Communication (1990) (as referenced in 97).

105. Aidun, C. K. Fundamentals of Coating Systems. Report to Engineering Project Advisory Committee, The Institute of Paper Chemistry, 1989.

106. Pranckh, F. R. and Scriven, L. E. The Physics of Blade Coating of Deformable Substrate. Proceedings TAPPI Coating Conference, TAPPI Press, Atlanta, GA p. 217, 1988.

107. Pranckh, F. R. and Scriven, L. E. Elastohydrodynamics of Blade Coating. AIChE J. 36:587(1990).

108. Deibler, J. A. and Cerro, R. L. Viscous Flow with a Free Surface Inside a Horizontal Rotating Drum. I. Hydrodynamics. Ind. Eng. Chem. Fund. 15(2):102-110(1976).

109. Orr, F. M. and Scriven, L. E. Rimming Flow: Numerical Simulation of Steady, Viscous, Free-Surface Flow with Surface Tension. J. Fluid Mech. 84(1):145-165(1978).

110. Wilhelmsson, B. I. and Stenström, S. G. University of Lund, Sweden, (Private communication of unpublished results).

111. Pulkowsi, J. K. and Wedel, G. L. The Effect of Spoiler Bars on Dryer Heat Transfer. Pulp and Paper Canada 89(8):61-66(1988).

112. Leonard, B. P.; A Stable, Accurate, Economical, and Comprehensible Algorithm for the Navier-Stokes and Scalar Transport Equations. Numerical Methods in Laminar and Turbulent Flow, 2nd International Conference, Venice, Italy, 543-54(1981).

113. Freitas, C. J.; Street, R. L.; Findikakis, A. N.; Koseff, J. R.; Numerical Simulations of Three-Dimensional Flow in a Cavity. Int. J. Num. Meth. Fluids, 5:561-75(1985).

114. Gerald, C. F. Applied Numerical Analysis. Second Edition, Addison-Wesley Publishing Company, Inc., 1980.

APPENDIX I

DESCRIPTION OF THE PROGRAM IPST-VOF3D

This appendix describes the operation of the IPST-VOF3D computational fluid dynamics program. This process will occur in four major steps. First, the general execution procedure of the program is outlined with an accompanying flow diagram. Next, the individual subroutines included in or called by the program are listed with a brief description, a list of the routines that call the subroutine, a list of the subroutines that are called by the subroutine, and a list of any files that are included in the subroutine. Third, the variables in common are documented (excluding any variables used as input data) with separate sections for the array and the scalar variables. This documentation includes a list of subroutines that modify the variable, other routines that use the variable, and the common include file and common block where the variable is defined. Finally, the input data are described with separate sections for each namelist. The input data are documented as to their modification, use, and default values.

OUTLINE OF PROGRAM EXECUTION

The IPST-VOF3D program consists of two major pieces as shown in Figure 1. Program execution begins in a primary controlling routine (**CONTROL**) which opens input and output files, calls initialization routines, and calls the main calculation routine **SOLA**.

Within **SOLA** execution continues in the same general manner as that outlined in the description of the numerical method found earlier. First, and explicit guess for the new velocity field is computed using either **TILDE**, **THIRD**, **QUICK**, or **KANDK**.

Next, after imposition of the boundary conditions, the pressure equation is solved using either the Conjugate Residual technique (**PRESCR**) or an SOR solver (**PRESSIT**). Both of these routines call **VISC3D** to compute the deviatoric normal stress at the interface as required. In addition **PRESSIT** calls **VAPOR1** to compute the pressure in the vapor phase at the interface if needed (at the present time the conjugate residual solver will not work if the vapor phase option is used).

After the boundary conditions have again been imposed, the VOF function is convected through the domain yielding a new surface configuration. This is accomplished using a form of donor-acceptor differencing to maintain a sharp interface between the two fluids. The boundary conditions are again updated and the surface physics routines are called.

The surface physics routines, overseen by **PETACAL** includes **PCAL** to compute preliminary surface orientations (calls **LAVORE** to identify separated void regions as needed), **PETASET** to compute the pressure interpolation factors, **PRESCK** to reset the pressure in all non-fluid cells, and either **SURCART** (Cartesian coordinates) or **SURF10N** (cylindrical coordinates) to compute the surface tension force as needed. The surface physics related task is solution of the vapor phase potential which, when required, is accomplished (after the boundary conditions are updated) by either **VAPOR** (two-dimensions) or **VAPOR3D** (three-dimensions). Next, required output prints and plots are performed, the values are advanced to the next time step, and the time and cycle are incremented. This completes the main calculation loop and the process begins again.

Several details of the solution procedure have been omitted in the interest of clarity. For instance, there is an initial startup section in **SOLA** that includes **VFCONV**, the surface physics routines in **PETACAL**, and the vapor phase routines **VAPOR** or

VAPOR3D as needed. The details of the algorithms used in the pressure equation solvers (**PRESCR** and **PRESSIT**), vapor phase solvers (**VAPOR** or **VAPOR3D**), and the surface tension subroutines (**SURF10N** or **SURCART**) can be found by looking at the inline documentation contained in each subroutine.

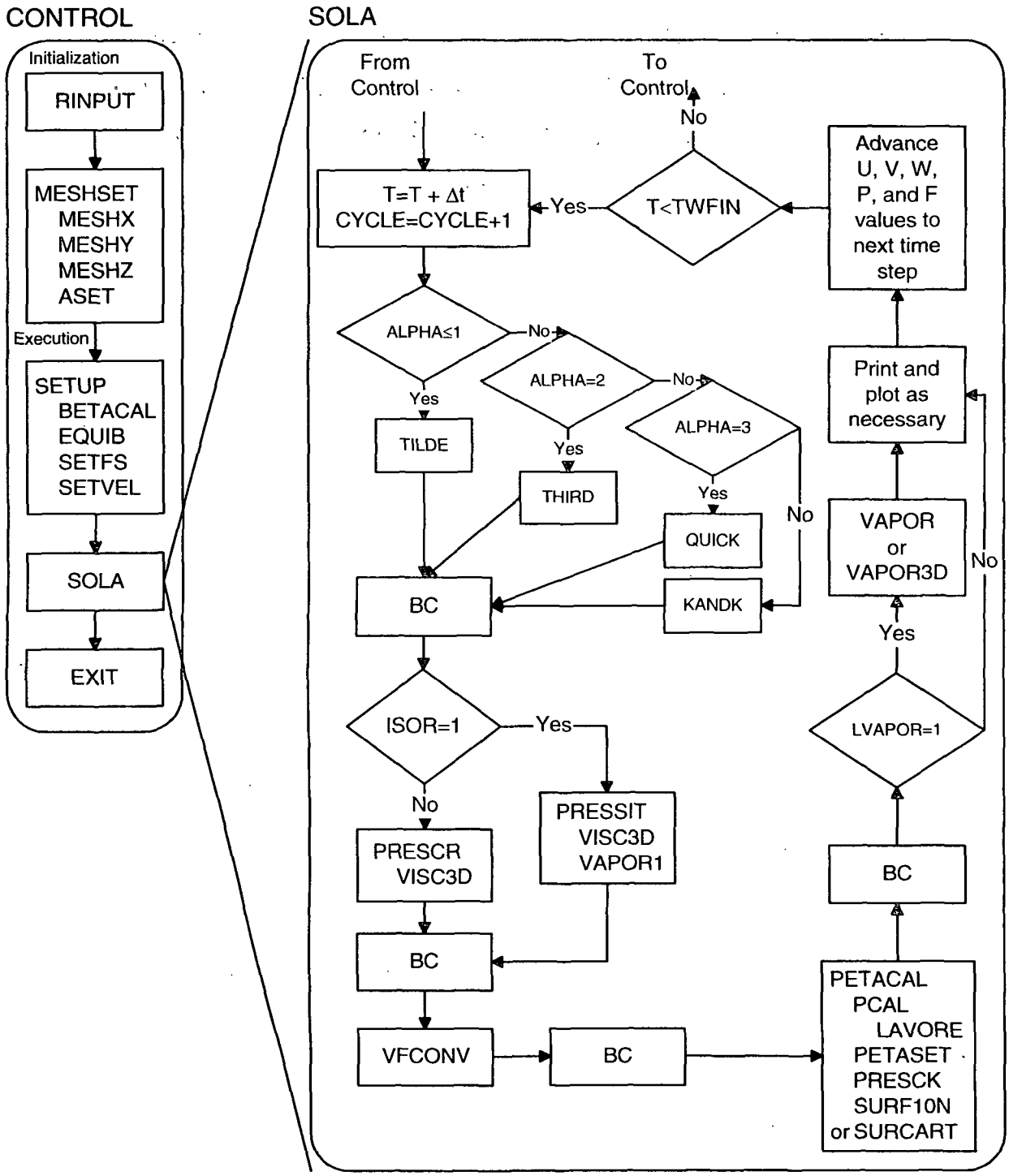

Figure I-1. Flow chart for IPST-VOF3D.

SUBROUTINE DOCUMENTATION

This section contains documentation of the subroutines that make up the program IPST-VOF3D. Each subroutine is documented by a brief description of its purpose, a list of the subroutines that call it, a list of the subroutines that it calls, and any files that must be available to be included at compile time.

AFACE1 (2D vapor phase) Treats the case where only one face is closed to the vapor phase.
Called by **VAPOR**
Calls **MATRIX2**
includes none
Added by John McKibben at IPST

AFACE2A (2D vapor phase) Treats the case where two adjacent faces are closed to the vapor phase.
Called by **VAPOR**
Calls **MATRIX2**
includes none
Added by John McKibben at IPST

AFACE2B (2D vapor phase) Treats the case where two opposing faces are closed to the vapor phase.
Called by **VAPOR**
Calls **MATRIX2**
includes none
Added by John McKibben at IPST

AFACE3 (2D vapor phase) Treats the case where three faces are closed to the vapor phase.
Called by **VAPOR**
Calls **MATRIX2**
includes none
Added by John McKibben at IPST

ASET (geometry setup) Sets up interior obstacles. Obstacle data read from namelist MESHGN in **MESHSET**
Called by **MESHSET**
Calls none
Includes 'vof3dcom'

BC (boundary conditions) Sets boundary conditions at the edges of the computational domain. Slip (symmetry), no-slip, continuative outflow, and periodic options are available. Calls BCFS to update the boundary conditions at the free surface. Modifications should be made in the patch file bc.pat.
Called by **PRESCR, PRESSIT, SOLA**
Calls **BCFS**
Includes 'vof3dcom' and 'bc.pat'

BCFS (boundary conditions) Sets the velocities in the cells adjacent to the free surface.
Called by **BC**
Calls none
Includes 'vof3dcom'

BETACAL (β for the mesh interior) Calculates BETA for non-obstacle cells. (Includes effects of over-relaxation parameter ω when SOR is used.)
Called by **SETUP**
Calls none
Includes 'vof3dcom'

CONTROL (controlling main program) Performs setup operations and reads restart tape as necessary. Main computations are performed under the control of **SOLA**. Normal program completion is accomplished here.
Called by This is the MAIN program
Calls **RINPUT, MESHSET, SETUP, SOLA, EXITA,**
Includes 'vof3dcom'

CPUTIME (utility) Gets the cpu time used by the program thus far.
Called by **RDTAPE** and **SOLA**
Calls system dependent functions
Includes none
Added by John McKibben at IPST

DATIM (utility) Gets the current date and time in the array NOW.
Called by **MESHSET** and **RDTAPE**
Calls **LTIME_** (system dependent function)
Includes none
Added by John McKibben at IPST

DCOPY (BLAS) Basic Linear Algebra Subroutine to copy one double precision vector to another.
Called by **DELTADJ**, **SETVEL**, and **SOLA**
Calls none
Includes none

DELTADJ (time step control) Adjusts the time step, if enabled with the AUTOT flag, to ensure numerical stability and a reasonable number of iterations for the pressure equations. Recomputes BETA as needed if SOR is used.
Called by **SOLA**
Calls **DCOPY**
Includes 'vof3dcom'

DRAW (graphics output) Replaces the graphics subroutines in the NASA-VOF3D program with a dump to an ASCII data file in the format needed by the Data Visualizer™ program. The data output are grid point locations XI(i), YJ(j), and ZK(k); the obstacle data AC(i,j,k); the VOF function F; the velocities at the center of each cell $U_{ave}(i,j,k)$, $V_{ave}(i,j,k)$, and $W_{ave}(i,j,k)$; the pressure P(i,j,k); and the vapor phase potential PRV(i,j,k). Subsets of the data can be output by changes in the patch file 'draw.pat'.
Called by , **SOLA**, **VAPOR**, and **VAPOR3D**
Calls **SYSTEM** (system dependent)
Includes 'vof3dcom', 'draw.pat' and 'vaporcom'
Rewritten by John McKibben at IPST

DSWAP (BLAS) Basic Linear Algebra Subroutine to swap two double precision vectors.
Called by **SOLA**
Calls none
Includes none

EQUIB (equilibrium surface) Solves two-point boundary value problem to obtain (cylindrically symmetric) equilibrium shape of initial free surface and returns to **SETUP**.
Called by **SETUP**
Calls **EXITA**
Includes none

EXITA (utility) Prints message and aborts the program.
Called by **CONTROL, EQUIB, PRESCR, RDTAPE, VAPOR, VAPOR3D,** and **VFCONV**
Calls none
Includes none
Added by John McKibben at IPST

FACE1 (3D vapor phase) Treats case where only one face is closed to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX**
Includes none
Added by John McKibben at IPST

FACE2A (3D vapor phase) Treats case where two adjacent faces are closed to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX**
Includes none
Added by John McKibben at IPST

FACE2B (3D vapor phase) Treats case where two opposing faces are closed to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX**
Includes none
Added by John McKibben at IPST

FACE3A (3D vapor phase) Treats case where three adjacent faces (forming a corner) are closed to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX**
Includes none
Added by John McKibben at IPST

FACE3B (3D vapor phase) Treats case where two opposing faces and one connecting face are closed to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX2** and **MATRIX**
Includes none
Added by John McKibben at IPST

FACE3C (3D vapor phase) Treats case where two opposing faces and one connecting face are closed to the vapor phase. (different orientation than FACE3B)
Called by **VAPOR3D**
Calls **MATRIX2** and **MATRIX**
Includes none
Added by John McKibben at IPST

FACE4A (3D vapor phase) Treats case where only two adjoining faces are open to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX2** and **MATRIX**
Includes none
Added by John McKibben at IPST

FACE4B (3D vapor phase) Treats case where only two opposing faces are open to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX2**
Includes none
Added by John McKibben at IPST

FACE5 (3D vapor phase) Treats case where only one face is open to the vapor phase.
Called by **VAPOR3D**
Calls **MATRIX**
Includes none
Added by John McKibben at IPST

KANDK (Provisional velocity field) Replacement for **TILDE** to compute the provisional velocity field using an approximate variable grid version of Kawamura and Kuwahara's third order accurate technique for treating the convective terms in the Navier-Stokes equations. Called if ALPHA = 4.0. (The 'tilde1.pat' and 'tilde2.pat' files are used to treat the singularity at the static contact line.)
Called by **SOLA**
Calls none
Includes 'vof3dcom', 'tilde1.pat', and 'tilde2.pat'
Added by John McKibben at IPST

LAVORE (vapor phase identification) Algorithm, modified from the NASA-VOF2D program to define each separate vapor phase region with its one NF value greater than 7. Needed so that the potential in each vapor phase region is treated separately.
Called by **PCAL**
Calls none
Includes 'vof3dcom'
Added by John McKibben at IPST

LPRT (output) Performs specified output of data to files. Options are determined by the flag LPR.
Called by **SOLA**
Calls none
Includes 'vof3dcom'

LPRT2 (output) Entry point in **LPRT**.
Called by **PRESSIT** and **SOLA**
Calls none
Includes 'vof3dcom'

LUBKSB (utility) From Numerical Recipes. Solves a system of equations by back-substitution with the LU decomposition.
Called by **MATRIX**
Calls none
Includes none
Added by John McKibben at IPST

LUDCMP (utility) From Numerical Recipes. Computes matrix LU decomposition.
Called by **MATRIX**
Calls none
Includes none
Added by John McKibben at IPST

MATRIX (vapor phase) Compute coefficients for Neuman boundary condition on a curved boundary given the location of the points and the normal vector at the surface. Uses points in three-dimensions and calls subroutines from Numerical Recipes to invert the matrix.
Called by **FACE1**, **FACE2A**, **FACE2B**, **FACE3A**, **FACE3B**, **FACE3C**, **FACE4A**, and **FACE5**
Calls **LUDCMP** and **LUBKSB**
Includes none
Added by John McKibben at IPST

MATRIX2 (vapor phase) Compute coefficients for Neuman boundary condition on a curved boundary given the location of the points and the normal vector at the surface. Uses points in two-dimensions and inverts matrices analytically.

Called by **AFACE1**, **AFACE2A**, **AFACE2B**, **AFACE3**, **FACE3B**, **FACE3C**, **FACE4A**, and **FACE4B**
Calls none
Includes none
Added by John McKibben at IPST

MESHSET (mesh generator) Generates computing mesh, the fractional volumes and areas open to flow, and geometric information about the mesh. Reads in the mesh and obstacle data from namelist MESHGN.

Called by **CONTROL**
Calls **ASET**, **MESHX**, **MESHY**, **MESHZ**, and **DATIM**
Includes 'vof3dcom'

MESHX (mesh generator) Computes x-coordinate values and their reciprocals.

Called by **MESHSET**
Calls none
Includes 'vof3dcom'

MESHY (mesh generator) Computes y-coordinate values and their reciprocals. Includes special treatment of case with only one non-"fictitious" cell.

Called by **MESHSET**
Calls none
Includes 'vof3dcom'

MESHZ (mesh generator) Computes z-coordinate values and their reciprocals.

Called by **MESHSET**
Calls none
Includes 'vof3dcom'

PCAL (surface physics) Calculates preliminary values of NF array and supplies them to **PETACAL**.

Called by **PETACAL**
Calls **LAVORE**
Includes 'vof3dcom'

PETACAL (surface physics) Determines preliminary values of surface orientation index NF and of interpolation parameter η (PETA).
Called by **SOLA**
Calls **PETASET, PCAL, PRESCK, SURCART** and **SURF10N**
Includes 'vof3dcom'

PETASET (surface physics) Calculates PETA in neighboring interpolation cells.
Called by **PETACAL**
Calls none
Includes 'vof3dcom'

PRESCK (surface physics) Resets pressure in surface cells, obstacle cells, void cells and isolated fluid cells.
Called by **PETACAL**
Calls none
Includes 'vof3dcom'

PRESCR (conjugate residual) Increments pressures and velocities by conjugate residual technique, as in NASA-VOF2D and NASA-VOF3D programs. Currently does not work when the vapor phase option is turned on.
Called by **SOLA**
Calls **BC, EXITA**, and **VISC3D**
Includes 'vof3dcom' and 'vaporcom'

PRESSIT (successive-over-relaxation) Increments pressures and velocities by successive-over-relaxation technique
Called by **SOLA**
Calls **BC, LPRT2, VAPOR1**, and **VISC3D**
Includes 'vof3dcom'

QUICK (Provisional velocity field) Replacement for **TILDE** to compute the provisional velocity field using a variable grid version of Leonard's Quadratic Upstream Interpolation for Convective Kinematics (QUICK) technique for treating the convective terms in the Navier-Stokes equations. Called if ALPHA = 3.0. (The 'tilde1.pat' and 'tilde2.pat' files are used to treat the singularity at the static contact line.)
Called by **SOLA**
Calls none
Includes 'vof3dcom', 'tilde1.pat', and 'tilde2.pat'
Added by John McKibben at IPST

RDTAPE (read restart tape) Reads restart data from tape.
Called by **SOLA**
Calls **CPUTIME, DATIM, EXITA,** and **RINPUT**
Includes 'vof3dcom'

RINPUT (read input data) Sets default values, reads input data from namelist XPUT, and echoes input data to output file.
Called by **CONTROL** and **RDTAPE**
Calls none
Includes 'vof3dcom'

SETFS (initial conditions) Generates initial configuration of fluid from data in namelist FLUIDGN.
Called by **SETUP**

SETUP (initial conditions) Calculates problem parameters, geometry, initial arrays, fluid configuration, and velocities. User modifications in the initial velocity fields are incorporated through 'setup.pat' and modifications to the initial fluid configuration are incorporated through 'setup1.pat'.
Called by **CONTROL**
Calls **BETACAL, EQUIB, SETFS,** and **SETVEL**
Includes 'vof3dcom', 'setup.pat', and 'setup1.pat'

SETVEL (initial conditions) Additional subroutine to set up the perturbations needed in instability studies. This also ensures that the FN array is filled in to ensure proper startup of vapor phase computations.
Called by **SETUP**
Calls **DCOPY**
Includes 'vof3dcom'
Added by John McKibben at IPST

SOLA (main calculational routine) Increments pressures, velocities, volume of fluid, and vapor phase potential by one time step δt.
Called by **CONTROL**
Calls **BC, CPUTIME, DCOPY , DELTADJ, DRAW, DSWAP, KANDK, LPRT, LPRT2, PETACAL, PRESCR, PRESSIT, QUICK, RDTAPE, THIRD, TILDE, VAPOR, VAPOR3D, VFCONV,** and **WRTAPE**
Includes 'vof3dcom'

SURCART (surface physics) Gives final NF values and calculates surface tension effect for Cartesian coordinates only (cylindrical coordinates are handled in **SURF10N**). Appropriate place for wall adhesion modifications. Patches needed for static contact points are included from 'surcart.pat'.
Called by **PETACAL**
Calls none
Includes 'vof3dcom' and 'surcart.pat'
Added by John McKibben at IPST

SURF10N (surface physics) Gives final NF values and calculates surface tension effect for cylindrical coordinates only (Cartesian coordinates are handled in **SURCART**). Appropriate place for wall adhesion modifications.
Called by **PETACAL**
Calls none
Includes 'vof3dcom'

THIRD (Provisional velocity field) Replacement for **TILDE** to compute the provisional velocity field using an variable grid version of Agarwal's third order accurate upwind differencing technique for treating the convective terms in the Navier-Stokes equations. Called if ALPHA = 2.0. (The 'tilde1.pat' and 'tilde2.pat' files are used to treat the singularity at the static contact line.)
Called by **SOLA**
Calls none
Includes 'vof3dcom', 'tilde1.pat', and 'tilde2.pat'
Added by John McKibben at IPST

TILDE (Provisional velocity field) Calculates explicitly a set of approximate velocity increments. Used if $0.0 \leq \text{ALPHA} \leq 1.0$, otherwise **THIRD**, **QUICK**, or **KANDK** is used. (The 'tilde1.pat' and 'tilde2.pat' files are used to treat the singularity at the static contact line.)
Called by **SOLA**
Calls none
Includes 'vof3dcom', 'tilde1.pat', and 'tilde2.pat'

VAPOR (2D vapor phase) Computes the vapor phase velocity potential for two-dimensional problems with Neuman Boundary conditions on curved boundaries using the method of Bramble and Hubbard. Loops through each vapor phase region based on the identifying region number from **LAVORE** at the reference location.

Called by **SOLA**
Calls **AFACE1, AFACE2A, AFACE2B, AFACE3, DRAW,** and **EXITA**
Includes 'vof3dcom' and 'vaporcom'
Added by John McKibben at IPST

VAPOR1 (surface physics) Computes the vapor phase pressure at the interface from the vapor phase potential computed in either **VAPOR** or **VAPOR3D**.

Called by **PRESSIT**
Calls none
Includes 'vof3dcom'
Added by John McKibben at IPST

VAPOR3D (3D vapor phase) Computes the vapor phase velocity potential for three-dimensional problems with Neuman Boundary conditions on curved boundaries using the method of Bramble and Hubbard. Loops through each vapor phase region based on the identifying region number from **LAVORE** at the reference location.

Called by **SOLA**
Calls **FACE1, FACE2A, FACE2B, FACE3A, FACE3B, FACE3C, FACE4A, FACE4B, FACE5, DRAW,** and **EXITA**
Includes 'vof3dcom' and 'vaporcom'
Added by John McKibben at IPST

VCHGCAL **(void volume)** Computes the volume of disjoint void (F = 0.0) regions.

Called by **VFCONV**
Calls none
Includes 'vof3dcom'

VFCONV (F increment) Computes the advective fluxes of F from the newly determined velocity field using Donor-Acceptor differencing and updates the F array. Patches needed for inlet boundaries (to ensure that fluid continues to enter the domain) are added in the file 'vfconv.pat.

Called by **SOLA**
Calls **VFCONV** and **EXITA**
Includes 'vof3dcom' and 'vfconv.pat'

VISC3D (surface physics) Computes the deviatoric stress component of the interfacial boundary condition. This is accomplished by computing the local rate of strain at the interface (using only velocities known within the liquid) and combining this with knowledge of the normal vector to compute $2\mu e_{ij}n_i n_j$.

Called by	**PRESSIT** and **PRESCR**
Calls	none
Includes	'vof3dcom'
Added by	John McKibben at IPST

WRTAPE (output) Writes restart data to output file.

Called by	**SOLA**
Calls	none
Include	'vof3dcom'

COMMUNICATION AMONG THE SUBROUTINES

In large part communication among the subroutines that make up IPST-VOF3D is accomplished through variables in common. The majority of these variables are contained in the file 'vof3dcom' referred to in the includes lists above. There are a few additional variables listed in 'vaporcom' which define a shared pool of temporary storage for use in **VAPOR**, **VAPOR3D**, **PRESCR**, and **DRAW**.

First I will describe the arrays in common (excluding input data) and their primary functions. This will be followed by a similar discussion of the scalar variables in common (excluding input data) and their functions. A few variables will occur both here and in the input data listing because these input variables are regularly updated as the simulation precedes.

Documentation consists of identification of the variables type, a brief description of its purpose, the subroutines where its value is modified, other subroutines that use it, and file and common block where it is defined.

Parameters in COMMON

IBASC (INTEGER) Maximum number of computational cells. Must be greater than IMAX*JMAX*KMAX
File 'vof3dcom'
Default = 100000

LSCR (INTEGER) Scratch storage space lscr=ibasc*maxcoef
File 'vaporcom'
Default = 1100000

MAXCOEF (INTEGER) Number of columns in "scratch" arrays
File 'vaporcom'
Default = 11

MXV (INTEGER) Maximum number of cells in the x-direction must be greater than IMAX
File 'vof3dcom'
Default = 500

MYV (INTEGER) Maximum number of cells in the x-direction must be greater than JMAX
File 'vof3dcom'
Default = 500

MZV (INTEGER) Maximum number of cells in the x-direction must be greater than KMAX
File 'vof3dcom'
Default = 500

NVOR (INTEGER) Maximum number of void regions. Set larger than expected due to needs of the region numbering algorithm in **LAVORE**.
File 'vof3dcom'
Default = 5000

Arrays in COMMON

ABK(i,j,k) (DOUBLE PRECISION) Fractional area open to flow in back wall of cell (i,j,k)
Modified in **ASET**
Used in **BCFS, BETACAL, KANDK, PCAL, PETASET, PRESCR, QUICK, SETUP, SURCART, SURF10N, THIRD, TILDE, VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM4/

AC(i,j,k) (DOUBLE PRECISION) Fractional volume open to flow in cell (i,j,k)
Modified in **ASET**
Used in **BETACAL, DRAW, KANDK, PCAL, PETASET, PRESCK, PRESCR, QUICK, SURCART, SURF10N, THIRD, VCHGCAL, VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM4/

AR(i,j,k) (DOUBLE PRECISION) Fractional area open to flow in right wall of cell (i,j,k)
Modified in **ASET**
Used in **BCFS, BETACAL, KANDK, PCAL, PETASET, PRESCR, QUICK, SETUP, SURCART, SURF10N, THIRD, TILDE, VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM4/

AT(i,j,k) (DOUBLE PRECISION) Fractional area open to flow in top wall of cell (i,j,k)
Modified in **ASET**
Used in **BCFS, BETACAL, PCAL, PETASET, PRESCR, SETUP, SURCART, SURF10N, VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM4/

BETA(i,j,k) (DOUBLE PRECISION) Pressure iteration relaxation factor in cell (i,j,k)
Modified in **ASET, BETACAL**, and **DELTADJ**
Used in **BCFS, KANDK, LPRT, PCAL, PETACAL, PETASET, PRESCK, PRESCR, PRESSIT, QUICK, SURCART, SURF10N, THIRD, TILDE, VCHGCAL**, and **VFCONV**
In common 'vof3dcom' /SLCM2/

CTHJ(j) (DOUBLE PRECISION) Cosine of θ (at cell center) = Cos(YJ(j)/X(IM1))
Modified in **MESHSET**
Used in **BC** and **SETUP**
In common 'vof3dcom' /SSCM4/

CTHJBK(j) (DOUBLE PRECISION) Cosine of θ (at cell back) = Cos(Y(j)/X(IM1))
Modified in **MESHSET**
Used in **BC**, **SETFS**, and **SETUP**
In common 'vof3dcom' /SSCM4/

D(i,j,k) (DOUBLE PRECISION) The residual ($\nabla \cdot \mathbf{u}$) for cell (i,j,k) after convergence of the pressure iteration
Modified in **PRESCR**, **PRESSIT**, and **SETUP**
Used in **VFCONV**
In common 'vof3dcom' /SLCM4/

DELX(i) (DOUBLE PRECISION) Mesh spacing of the i-th cell along the radial (x) coordinate
Modified in **MESHX**
Used in **ASET**, **DELTADJ**, **KANDK**, **MESHSET**, **PETACAL**, **PRESCR**, **QUICK**, **SETFS**, **SETUP**, **SURCART**, **SURF10N**, **THIRD**, **TILDE**, **VAPOR**, **VAPOR1**, **VAPOR3D**, **VCHGCAL**, **VFCONV**, and **VISC3D**
In common 'vof3dcom' /SSCM1/

DELY(j) (DOUBLE PRECISION) Mesh spacing of the j-th cell along the azimuthal (y) coordinate at the maximum R(x) mesh value
Modified in **MESHY**
Used in **DELTADJ**, **KANDK**, **MESHSET**, **PETACAL**, **PRESCR**, **QUICK**, **SETFS**, **SURCART**, **SURF10N**, **THIRD**, **TILDE**, **VAPOR1**, **VAPOR3D**, **VCHGCAL**, **VFCONV**, and **VISC3D**
In common 'vof3dcom' /SSCM1/

DELZ(k) (DOUBLE PRECISION) Mesh spacing of the k-th cell along the axial (z) coordinate
Modified in **MESHZ**
Used in **ASET**, **DELTADJ**, **KANDK**, **MESHSET**, **PETACAL**, **PRESCR**, **QUICK**, **SETFS**, **SETUP**, **SURCART**, **SURF10N**, **THIRD**, **TILDE**, **VAPOR**, **VAPOR1**, **VAPOR3D**, **VCHGCAL**, **VFCONV**, and **VISC3D**
In common 'vof3dcom' /SSCM1/

F(i,j,k) (DOUBLE PRECISION) Volume of fluid per unit volume of cell (i,j,k) at time level n+1
Modified in **BC, BCFS, DELTADJ, SETFS, SETUP, VCHGCAL,** and **VFCONV**
Used in **DRAW, KANDK, LPRT, PCAL, PETACAL, PRESCR, QUICK, SOLA, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SLCM2/

FN(i,j,k) (DOUBLE PRECISION) Volume of fluid per unit volume of cell (i,j,k) at time level n
Modified in **SETUP** and **SOLA**
Used in **DELTADJ, VAPOR, VAPOR3D,** and **VFCONV**
In common 'vof3dcom' /SLCM2/

GXA(j) (DOUBLE PRECISION) Radial component of acceleration due to applied body force for cell (i,j,k)
Modified in **SETUP**
Used in **KANDK, QUICK, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM5/

GYA(j) (DOUBLE PRECISION) Azimuthal component of acceleration due to applied body force for cell (i,j,k)
Modified in **SETUP**
Used in **KANDK, QUICK, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM5/

JOP(i) (INTEGER) Index value of plane opposite j-th J plate (appropriate for CYL = 1.0 only)
Modified in **MESHSET**
Used in **BC, PCAL,** and **SURF10N**
In common 'vof3dcom' /SSCM4A/

NF(i,j,k) (INTEGER) Flag indicating cell type at time level n+1
Modified in **BC, LAVORE, PCAL, PETACAL, SETUP,** and **SURF10N**
Used in **BCFS, DELTADJ, DRAW, KANDK, LPRT, PETASET, PRESCK, PRESCR, PRESSIT, QUICK, THIRD, VAPOR, VAPOR1, VAPOR3D, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SLCM3/

NFO(i,j,k) (INTEGER) Flag indicating cell type at time level n
Modified in **PETACAL**
Used in **LPRT** and **PRESCK**
In common 'vof3dcom' /SLCM3/

NFP(i,j,k) (INTEGER) Flag indicating provisional cell type at time level n+1 (from fluid surface height function)
Modified in **PCAL** and **PETACAL**
Used in **LPRT**
In common 'vof3dcom' /SLCM3/

NFS(i,j,k) (INTEGER) Flag indicating provisional cell type at time level n+1 (from fluid surface slope function)
Modified in **PETACAL, SURCART**, and **SURF10N**
Used in **LPRT**
In common 'vof3dcom' /SLCM3/

P(i,j,k) (DOUBLE PRECISION) Pressure in cell (i,j,k) at time level n+1 divided by the liquid phase density
Modified in **BC, BCFS, PETACAL, PETASET, PRESCK, PRESCR, PRESSIT**, and **SETUP**
Used in **DRAW, KANDK, LPRT, QUICK, SOLA, THIRD**, and **TILDE**
In common 'vof3dcom' /SLCM2/

PETA(i,j,k) (DOUBLE PRECISION) Pressure interpolation factor for cell (i,j,k)
Modified in **BC, PETACAL, PETASET**, and **SETUP**
Used in **LPRT, PRESCK, PRESCR**, and **PRESSIT**
In common 'vof3dcom' /SLCM2/

PN(i,j,k) (DOUBLE PRECISION) Pressure in cell (i,j,k) at time level n divided by the liquid phase density
Modified in **BC, PRESCR, SETUP, SOLA, SURCART**, and **SURF10N**
Used in **PRESCK**
In common 'vof3dcom' /SLCM2/

PR(NVOR) (DOUBLE PRECISION) Pressure in void region I (nominally = 0)
Modified in **SETUP**
Used in **PETASET, PRESCK, PRESCR**, and **PRESSIT**
In common 'vof3dcom' /SSCM4/

PS(i,j,k) (DOUBLE PRECISION) Surface pressure in cell (i,j,k) computed from surface tension forces
Modified in **BC, PETACAL, SURCART,** and **SURF10N**
Used in **LPRT, PRESCK, PRESCR,** and **PRESSIT**
In common 'vof3dcom' /SLCM4/

RDX(i) (DOUBLE PRECISION) Reciprocal of DELX(i)
Modified in **MESHX**
Used in **BCFS, BETACAL, KANDK, MESHSET, PETASET, PRESCR, PRESSIT, QUICK, SURF10N, THIRD, TILDE, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RDXP(i) (DOUBLE PRECISION) 2.0/(DELX(i)+DELX(i+1))
Modified in **MESHX**
Used in **BETACAL, DELTADJ, KANDK, PETASET, PRESCR, PRESSIT, QUICK, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RDY(j) (DOUBLE PRECISION) Reciprocal of DELY(j)
Modified in **MESHY**
Used in **BCFS, BETACAL, KANDK, MESHSET, PETASET, PRESCR, PRESSIT, QUICK, SURF10N, THIRD, TILDE, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RDYP(j) (DOUBLE PRECISION) 2.0/(DELY(j)+DELY(j+1))
Modified in **MESHY**
Used in **BETACAL, DELTADJ, KANDK, PETASET, PRESCR, PRESSIT, QUICK, THIRD, TILDE, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RDZ(k) (DOUBLE PRECISION) Reciprocal of DELZ(k)
Modified in **MESHZ**
Used in **BCFS, BETACAL, KANDK, MESHSET, PETASET, PRESCR, PRESSIT, QUICK, SETUP, SURF10N, THIRD, TILDE, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RDZP(k) (DOUBLE PRECISION) 2.0/(DELZ(k)+DELZ(k+1))
Modified in **MESHZ**
Used in **BETACAL, DELTADJ, KANDK, PETASET, PRESCR, PRESSIT, QUICK, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

RR(i) (DOUBLE PRECISION) =1.0 if CYL = 0.0; =X(IM1)/X(i) if CYL = 1.0
Modified in **MESHSET**
Used in. **BC, BCFS, KANDK, PRESCR, QUICK, THIRD, TILDE, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM6/

RRI(i) (DOUBLE PRECISION) =1.0 if CYL = 0.0; =X(IM1)/XI(i) if CYL = 1.0
Modified in **MESHSET**
Used in **BCFS, BETACAL, DELTADJ, KANDK, PETASET, PRESCR, PRESSIT, QUICK, SETFS, SURF10N, THIRD, TILDE, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM6/

RX(i) (DOUBLE PRECISION) Reciprocal of X(i)
Modified in **MESHX**
Used in **BETACAL, DRAW, KANDK, MESHSET, QUICK, SETUP, SURCART, SURF10N, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM2/

RXI(i) (DOUBLE PRECISION) Reciprocal of XI(i)
Modified in **MESHX**
Used in **BC, KANDK, MESHSET, QUICK, SURF10N, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM2/

RY(j) (DOUBLE PRECISION) Reciprocal of Y(j)
Modified in **MESHY**
Used in **SURCART**
In common 'vof3dcom' /SSCM2/

RYJ(j) (DOUBLE PRECISION) Reciprocal of YJ(j)
Modified in **MESHY**
Used in **MESHSET**
In common 'vof3dcom' /SSCM2/

RZ(k) (DOUBLE PRECISION) Reciprocal of Z(k)
Modified in **MESHZ**
Used in **SURCART**
In common 'vof3dcom' /SSCM2/

RZK(k) (DOUBLE PRECISION) Reciprocal of ZK(k)
Modified in **MESHZ**
Used in **MESHSET**
In common 'vof3dcom' /SSCM2/

STHJ(j) (DOUBLE PRECISION) Sin of θ (at cell center) =Sin(YJ(j)/X(IM1))
Modified in **MESHSET**
Used in **BC** and **SETUP**
In common 'vof3dcom' /SSCM4/

STHJBK(j) (DOUBLE PRECISION) Sin of θ (at cell back) =Sin(Y(j)/X(IM1))
Modified in **MESHSET**
Used in **BC**, **SETFS**, and **SETUP**
In common 'vof3dcom' /SSCM4/

U(i,j,k) (DOUBLE PRECISION) x-direction velocity in cell (i,j,k) at time level n+1
Modified in **BC**, **BCFS**, **DELTADJ**, **KANDK**, **PRESCR**, **PRESSIT**, **QUICK**, **SETUP**, **THIRD**, and **TILDE**
Used in **DRAW**, **LPRT**, **SOLA**, **VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM1/

UN(i,j,k) (DOUBLE PRECISION) x-direction velocity in cell (i,j,k) at time level n
Modified in **SOLA**
Used in **DELTADJ**, **KANDK**, **QUICK**, **THIRD**, and **TILDE**
In common 'vof3dcom' /SLCM1/

V(i,j,k) (DOUBLE PRECISION) y-direction velocity in cell (i,j,k) at time level n+1
Modified in **BC**, **BCFS**, **DELTADJ**, **KANDK**, **PRESCR**, **PRESSIT**, **QUICK**, **SETUP**, **THIRD**, and **TILDE**
Used in **DRAW**, **LPRT**, **SOLA**, **VFCONV**, and **VISC3D**
In common 'vof3dcom' /SLCM1/

VN(i,j,k) (DOUBLE PRECISION) y-direction velocity in cell (i,j,k) at time level n
Modified in **SOLA**
Used in **DELTADJ**, **KANDK**, **QUICK**, **THIRD**, and **TILDE**
In common 'vof3dcom' /SLCM1/

W(i,j,k) (DOUBLE PRECISION) z-direction velocity in cell (i,j,k) at time level n+1
Modified in **BC, BCFS, DELTADJ, KANDK, PRESCR, PRESSIT, QUICK, SETUP, THIRD,** and **TILDE**
Used in **DRAW, LPRT, SOLA, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SLCM1/

WN(i,j,k) (DOUBLE PRECISION) z-direction velocity in cell (i,j,k) at time level n
Modified in **SOLA**
Used in **DELTADJ, KANDK, QUICK, THIRD,** and **TILDE**
In common 'vof3dcom' /SLCM1/

X(i) (DOUBLE PRECISION) Location of the right-hand boundary of the i-th cell along the x-axis
Modified in **MESHX**
Used in **ASET, BCFS, KANDK, MESHSET, QUICK, SETFS, SETUP, SURF10N, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM2/

XI(i) (DOUBLE PRECISION) Location of the center of the i-th cell along the x-axis
Modified in **MESHX**
Used in **BC, DRAW, KANDK, MESHSET, QUICK, SETUP, SURF10N, THIRD, VAPOR, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

Y(j) (DOUBLE PRECISION) Location of the back boundary of the j-th cell along the y-axis
Modified in **MESHY**
Used in **MESHSET, SETFS,** and **SURF10N**
In common 'vof3dcom' /SSCM2/

YJ(j) (DOUBLE PRECISION) Location of the center of the j-th cell along the y-axis
Modified in **MESHY**
Used in **DRAW, KANDK, MESHSET, QUICK, SURF10N, THIRD, VAPOR1, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2/

Z(k) (DOUBLE PRECISION) Location of the top boundary of the k-th cell along the z-axis
Modified in **MESHZ**
Used in **ASET**, **MESHSET**, **SETFS**, **SETUP**, and **SURF10N**
In common 'vof3dcom' /SSCM2/

ZK(k) (DOUBLE PRECISION) Location of the center of the k-th cell along the z-axis
Modified in **MESHZ**
Used in **DRAW**, **KANDK**, **MESHSET**, **QUICK**, **THIRD**, **VAPOR**, **VAPOR1**, **VAPOR3D**, and **VISC3D**
In common 'vof3dcom' /SSCM2/

Additional variables in COMMON added by John F. McKibben at IPST

ASCR(i) (DOUBLE PRECISION) Scratch array for storing vapor phase potential coefficients in **VAPOR** and **VAPOR3D**, conjugate residual storage arrays in **PRESCR**, and temporary storage in **DRAW**.
Modified in **VAPOR** and **VAPOR3D**
Used in **DRAW** and **PRESCR**
In common 'vaporcom' /PRESVAP/

DIV(i,j,k) (DOUBLE PRECISION) Stores the divergence in **VAPOR**, **VAPOR3D**, and **PRESCR**
Modified in **PRESCR**, **VAPOR**, and **VAPOR3D**
Used in
In common 'vaporcom' /PRESVAP/

DPS(i,j,k) (DOUBLE PRECISION) Stores the correction to PS(i,j,k) due to the deviatoric stress (**VISC3D**) and the vapor phase pressure (**VAPOR1**)
Modified in **PRESCR**, **PRESSIT**, **VAPOR1**, and **VISC3D**
Used in **LPRT** and **PRESCK**
In common 'vof3dcom' /SLCM5/

DVMAX(n) (DOUBLE PRECISION) Maximum relative error in vapor region (n)
Modified in **VAPOR** and **VAPOR3D**
Used in **PRESSIT** and **SOLA**
In common 'vof3dcom' /MCKIB/

FNN(i,j,k) (DOUBLE PRECISION) Volume of fluid per unit volume of cell (i,j,k) at time level n-1
Modified in **SOLA**
Used in
In common 'vof3dcom' /SLCM2/

ISCR(i) (INTEGER) Scratch storage for pointer array for solution of vapor phase potential in **VAPOR** and **VAPOR3D**
Modified in **VAPOR** and **VAPOR3D**
Used in
In common 'vaporcom' /PRESVAP/

IMFL(ibasc) (INTEGER)
Modified in **PRESCR**, **VAPOR**, and **VAPOR3D**
Used in
In common 'vaporcom' /PRESVAP/

IVITER(n) (INTEGER) Iterations required for vapor region (n)
Modified in **VAPOR** and **VAPOR3D**
Used in **PRESSIT** and **SOLA**
In common 'vof3dcom' /MCKIB1/

NR(n) (INTEGER) Void region number
Modified in **LAVORE**
Used in
In common 'vof3dcom' /MCKIB1/

PRV(i,j,k) (DOUBLE PRECISION) Vapor phase potential of cell (i,j,k) at n+1
Modified in **VAPOR** and **VAPOR3D**
Used in **DRAW**, **PRESCR**, **SOLA**, and **VAPOR1**
In common 'vof3dcom' /SLCM5/

PRVN(i,j,k) (DOUBLE PRECISION) Vapor phase potential of cell (i,j,k) at n
Modified in **SOLA**, **VAPOR**, and **VAPOR3D**
Used in **PRESCR** and **VAPOR1**
In common 'vof3dcom' /SLCM5/

PRVNN(i,j,k) (DOUBLE PRECISION) Vapor phase potential of cell (i,j,k) at n-1
Modified in
Used in
In common 'vof3dcom' /SLCM5/

PRVT(ibasc) (DOUBLE PRECISION) Temporary storage array
Modified in **VAPOR** and **VAPOR3D**
Used in
In common 'vaporcom' /PRESVAP/

Scalar Variables in COMMON

AVE (DOUBLE PRECISION) Second-order accurate option parameter
Modified in **SOLA**
Used in **TILDE**
In common 'vof3dcom' /SSCM4/

CLK (DOUBLE PRECISION) System furnished time of day for run identification
CLK
Modified in **LPRT**
Used in **MESHSET**
In common 'vof3dcom' /SSCM1B/

CSANG (DOUBLE PRECISION) Cosine of contact angle
Modified in **RINPUT** and **SURF10N**
Used in **SURCART**
In common 'vof3dcom' /SSCM4/

CYCLE (INTEGER) Calculational time cycle number
Modified in **DELTADJ**, **SETUP**, and **SOLA**
Used in **LAVORE**, **LPRT**, **PETACAL**, **PRESCR**, **PRESSIT**, **VAPOR**, **VAPOR3D**, and **VFCONV**
In common 'vof3dcom' /SSCM1A/

DAT (CHARACTER*8) Date furnished by system for run identification
Modified in **LPRT**
Used in **MESHSET**
In common 'vof3dcom' /SSCM1B/

DELMN (DOUBLE PRECISION) Smallest cell dimension in problem
Modified in **MESHSET**
Used in
In common 'vof3dcom' /SSCM1/

DELT (DOUBLE PRECISION) Time step (also in input namelist XPUT)
Modified in **DELTADJ** and **RINPUT**
Used in **BETACAL, KANDK, LPRT, PETASET, PRESCR, PRESSIT, QUICK, SETUP, SOLA, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D,** and **VFCONV**
In common 'vof3dcom' /SSCM1/

DELXRL (DOUBLE PRECISION) Ratio of cell spacing on left side of mesh (=DELX(1)/DELX(2))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DELXRR (DOUBLE PRECISION) Ratio of cell spacing on right side of mesh (=DELX(IMAX)/DELX(IM1))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DELYRBK (DOUBLE PRECISION) Ratio of cell spacing on back side of mesh (=DELY(1)/DELY(2))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DELYRF (DOUBLE PRECISION) Ratio of cell spacing on front side of mesh (=DELY(JMAX)/DELY(JM1))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DELZRB (DOUBLE PRECISION) Ratio of cell spacing on bottom side of mesh (=DELZ(1)/DELZ(2))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DELZRT (DOUBLE PRECISION) Ratio of cell spacing on top side of mesh (=DELZ(KMAX)/DELZ(KM1))
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1/

DTVIS (DOUBLE PRECISION) Maximum DELT value allowed by the viscous forces stability criterion
Modified in **SETUP**
Used in **DELTADJ**
In common 'vof3dcom' /SSCM1/

EM6 (DOUBLE PRECISION) 10^{-6}
Modified in **ASET**
Used in **BC, BCFS, KANDK, PETASET, PRESCK, PRESCR, QUICK, RINPUT, SURCART, SURF10N, THIRD, TILDE, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM4/

EMF (DOUBLE PRECISION) Small value used to negate round off error effects in F convection ($=10^{-6}$)
Modified in **SETUP**
Used in **KANDK, PCAL, QUICK, SURCART, SURF10N, THIRD, TILDE, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM4/

EMF1 (DOUBLE PRECISION) 1.0-EMF
Modified in **SETUP**
Used in **PCAL, PRESCR,** and **VCHGCAL**
In common 'vof3dcom' /SSCM4/

FLG (DOUBLE PRECISION) Pressure iteration convergence indicator (=0.0 converged)
Modified in **PRESSIT, SETUP,** and **SOLA**
Used in **BCFS**
In common 'vof3dcom' /SSCM4/

FLGC (DOUBLE PRECISION) Volume of fluid convection limit indicator
Modified in **SETUP** and **VFCONV**
Used in **DELTADJ** and **SOLA**
In common 'vof3dcom' /SSCM4/

FNOC (DOUBLE PRECISION) Pressure convergence failure indication
Modified in **PRESSIT, SETUP** and **SOLA**
Used in **DELTADJ**
In common 'vof3dcom' /SSCM4/

IBAR (INTEGER) Number of real cells in x-direction (excludes fictitious cells)
Modified in **MESHX**
Used in **MESHSET, THIRD, VAPOR,** and **VAPOR3D**
In common 'vof3dcom' /SSCM1A/

II0 (INTEGER) Data storage parameter (=IBAR)
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1A/

II1 (INTEGER) Data storage parameter (=IMAX)
Modified in **MESHSET**
Used in **BCFS, PCAL, SURCART, SURF10N, TILDE, VAPOR1,** and **VAPOR3D, VISC3D**
In common 'vof3dcom' /SSCM1A/

II2 (INTEGER) Data storage parameter (=IMAX*JMAX)
Modified in **MESHSET**
Modified in **MESHSET**
Used in **BCFS, PCAL, SURF10N, THIRD, TILDE,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

II3 (INTEGER) Data storage parameter (=IMAX*JBAR)
Modified in **MESHSET**
Used in **BC**
In common 'vof3dcom' /SSCM1A/

II4 (INTEGER) Data storage parameter (=IMAX*JMAX*KBAR)
Modified in **MESHSET**
Used in **BC, PRESCR, VAPOR,** and **VAPOR3D**
In common 'vof3dcom' /SSCM1A/

II5 (INTEGER) Data storage parameter (=IMAX*JMAX)
Modified in **MESHSET**
Used in **ASET, BC, BCFS, DELTADJ, DRAW, KANDK, LAVORE, LPRT, PCAL, PETACAL, PETASET, PRESCK, PRESCR, PRESSIT, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

II6 (INTEGER) Data storage parameter (=IMAX*JBAR)
Modified in **MESHSET**
Used in **ASET** and **PRESCR**
In common 'vof3dcom' /SSCM1A/

II7 (INTEGER) Data storage parameter (=IMAX*JM1)
Modified in **MESHSET**
Used in **ASET, DRAW,** and **PRESCR**
In common 'vof3dcom' /SSCM1A/

IM1 (INTEGER) Value of the index I at the last real cell in the x-direction (=IMAX-1)
Modified in **MESHSET**
Used in **ASET, BC, BCFS, BETACAL, DELTADJ, DRAW, KANDK, LAVORE, PCAL, PETACAL, PRESCK, PRESCR, PRESSIT, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

IM2 (INTEGER) Value of the index I at the next to the last real cell in the x-direction (=IMAX-2)
Modified in **MESHSET**
Used in **KANDK, QUICK, SURCART, SURF10N, VAPOR,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

IMAX (INTEGER) Total number of cells in x-direction (=IBAR+2)
Modified in **MESHSET**
Used in **ASET, BC, BCFS, BETACAL, DELTADJ, DRAW, KANDK, LAVORE, LPRT, PCAL, PETACAL, PETASET, PRESCK, PRESCR, PRESSIT, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

ITER (INTEGER) Pressure iteration counter
Modified in **PRESCR, PRESSIT, SETUP,** and **SOLA**
Used in **BC, BCFS, DELTADJ, LPRT,** and **VFCONV**
In common 'vof3dcom' /SSCM1A/

JBAR (INTEGER) Number of real cells in y-direction (=JMAX-2)
Modified in **MESHY**
Used in **BC, MESHSET, SURF10N, THIRD,** and **VAPOR3D**
In common 'vof3dcom' /SSCM1A/

JC2PI (INTEGER) Indicator of 360 degree geometry (CYL = 1.0)
Modified in **MESHSET**
Used in **SURF10N**
In common 'vof3dcom' /SSCM4A/

JM1 (INTEGER) Value of the index J at the last real cell in the y-direction (JMAX-1)
Modified in **MESHSET**
Used in **ASET, BC, BCFS, BETACAL, DELTADJ, DRAW, KANDK, LAVORE, PCAL, PETACAL, PRESCK, PRESCR, PRESSIT, QUICK, SETUP, SOLA, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

JM2 (INTEGER) Value of the index J at the next to the last real cell in the y-direction (JMAX-2)
Modified in **MESHSET**
Used in **KANDK, QUICK, SURCART, SURF10N, VAPOR, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

JMAX (INTEGER) Total number of cells in y-direction (=JBAR+2)
Modified in **MESHSET**
Used in **ASET, BC, DRAW, KANDK, LAVORE, LPRT, PCAL, PETASET, PRESCR, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, VAPOR, VAPOR3D,** and **VFCONV**
In common 'vof3dcom' /SSCM1A/

KBAR (INTEGER) Number of real cells in z-direction (=KMAX-2)
Modified in **MESHZ**
Used in **MESHSET, THIRD, VAPOR,** and **VAPOR3D**
In common 'vof3dcom' /SSCM1A/

KM1 (INTEGER) Value of the index K at the last real cell in the z-direction (KMAX-1)
Modified in **MESHSET**
Used in **ASET, BCFS, BETACAL, DELTADJ, DRAW, KANDK, LAVORE, PCAL, PETACAL, PRESCK, PRESCR, PRESSIT, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D, VCHGCAL, VFCONV,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

KM2 (INTEGER) Value of the index K at the next to the last real cell in the z-direction (KMAX-2)
Modified in **MESHSET**
Used in **ASET, KANDK, QUICK, SURCART, SURF10N,** and **VISC3D**
In common 'vof3dcom' /SSCM1A/

KMAX (INTEGER) Total number of cells in z-direction (=KBAR+2)
Modified in **MESHSET**
Used in **ASET, B, DRAW, KANDK, LPRT, PCAL, PETASET, PRESCR, QUICK, SETFS, SETUP, SURCART, SURF10N, THIRD, VAPOR, VAPOR3D,** and **VFCONV**
In common 'vof3dcom' /SSCM1A/

NFLGC (INTEGER) Accumulated F convection limit excesses
Modified in **DELTADJ** and **SETUP**
Used in **SOLA**
In common 'vof3dcom' /SSCM4A/

NOCON (INTEGER) Accumulated pressure convergence failures
Modified in **PRESSIT** and **SETUP**
Used in **SOLA**
In common 'vof3dcom' /SSCM4A/

NUMTD (INTEGER) Restart tape dump counter
Modified in **SETUP**
Used in
In common 'vof3dcom' /SSCM2A/

PI (DOUBLE PRECISION) =3.141592654
Modified in **SETUP**
Used in
In common 'vof3dcom' /SSCM4/

RIJK (DOUBLE PRECISION) Reciprocal of the number of real non-obstacle cells on computational mesh
Modified in **BETACAL**
Used in **SOLA**
In common 'vof3dcom' /SSCM2/

SANG (DOUBLE PRECISION) Sine of the contact angle
Modified in **RINPUT** and **SURF10N**
Used in **SURCART**
In common 'vof3dcom' /SSCM4/

STIM (DOUBLE PRECISION) System clock time when problem commences
Modified in **SOLA**
Used in
In common 'vof3dcom' /SSCM2/

T (DOUBLE PRECISION) Current program time (initialized in input namelist XPUT)
Modified in **DELTADJ, RINPUT, SETUP,** and **SOLA**
Used in **DRAW, LPRT,** and **VFCONV**
In common 'vof3dcom' /SSCM2/

TANCA (DOUBLE PRECISION) Tangent of contact angle
Modified in **RINPUT**
Used in
In common 'vof3dcom' /SSCM4/

TD (INTEGER) Tape dump number (initialized in input namelist XPUT)
Modified in **RINPUT**
Used in **SOLA**
In common 'vof3dcom' /SSCM2A/

TLM (CHARACTER*8) Problem control parameter (currently not used)
Modified in
Used in
In common 'vof3dcom' /SSCM2/

TWPLT (DOUBLE PRECISION) Problem time for next plot and/or data print to be sent to film
Modified in **SETUP** and **SOLA**
Used in
In common 'vof3dcom' /SSCM2/

TWPRT (DOUBLE PRECISION) Problem time for next paper data print to be sent to output
Modified in **SETUP** and **SOLA**
Used in
In common 'vof3dcom' /SSCM2/

TWTD (DOUBLE PRECISION) Problem time for next restart tape dump
Modified in **SOLA**
Used in
In common 'vof3dcom' /SSCM2/

UDUM (DOUBLE PRECISION) Temporary storage for U component of velocity
Modified in **TILDE**
Used in
In common 'vof3dcom' /SSCM4/

VCHGT (DOUBLE PRECISION) Accumulated fluid volume change in mesh
Modified in **SETUP** and **VCHGCAL**
Used in **SOLA**
In common 'vof3dcom' /SSCM4/

VDUM (DOUBLE PRECISION) Temporary storage for V component of velocity
Modified in **TILDE**
Used in
In common 'vof3dcom' /SSCM4/

VOFTOT (DOUBLE PRECISION) Total fluid volume change in mesh
Modified in **VCHGCAL**
Used in **SOLA**
In common 'vof3dcom' /SSCM4/

WDUM (DOUBLE PRECISION) Temporary storage for W component of velocity
Modified in **TILDE**
Used in
In common 'vof3dcom' /SSCM4/

Additional variables in COMMON added by John F. McKibben at IPST

DELTN (DOUBLE PRECISION) Old value of DELT
Modified in **DELTADJ** and **RINPUT**
Used in **PRESCR, VAPOR, VAPOR1,** and **VAPOR3D**
In common 'vof3dcom' /MCKIB/

DELTNN (DOUBLE PRECISION) Old value of DELT
Modified in **DELTADJ** and **RINPUT**
Used in
In common 'vof3dcom' /MCKIB/

LVEC (INTEGER) Total number of computational cells in simulation
=IMAX*JMAX*KMAX
Modified in **CONTROL**
Used in **DELTADJ, DRAW, PETACAL, PRESCR, PRESSIT, SETUP, SOLA, SURCART, VAPOR,** and **VAPOR3D**
In common 'vof3dcom' /MCKIB1/

NAMELIST DOCUMENTATION

Variables in Namelist XPUT read in **RINPUT**

ALPHA (DOUBLE PRECISION) Controls the accuracy of the differencing of the convective terms in the Navier-Stokes equation.
0≤ALPHA≤1 Determines the portion of first order accurate upwind differencing used in **TILDE**.
ALPHA = 2 Uses third order accurate upwind differencing in **THIRD**.
ALPHA = 3 Uses quadratic upstream interpolation for convective kinematics in **QUICK**.
ALPHA = 4 Uses an approximate variable grid version of Kawamura and Kuwahara's differencing scheme in **KANDK**.
Modified in **RINPUT, SETUP,** and **SOLA**
Used in **TILDE**
In common 'vof3dcom' /SSCM1/
Default = 1.0

AUTOT (DOUBLE PRECISION) Automatic time stepping flag
AUTOT = 0.0 Use constant time step.
AUTOT = 1.0 Automatically adjust time step to maintain stability and a reasonable number of iterations for the pressure equation.
Modified in **RINPUT**
Used in **DELTADJ**
In common 'vof3dcom' /SSCM1/
Default = 1.0

CANGLE (DOUBLE PRECISION) Contact angle in degrees between the fluid and a wall.
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM4/
Default = 0.0

CYL (DOUBLE PRECISION) Cylindrical coordinates flag
CYL = 0.0 Use Cartesian coordinates.
CYL = 1.0 Use cylindrical coordinates.
Modified in **RINPUT** and **SETUP**
Used in **BC, BCFS, BETACAL, DRAW, KANDK, MESHSET, PCAL, PETACAL, QUICK, THIRD, TILDE,** and **VISC3D**
In common 'vof3dcom' /SSCM1/
Default = 0.0

DELT (DOUBLE PRECISION) Initial time step. Becomes the transient time step later.
Modified in **DELTADJ** and **RINPUT**
Used in **BETACAL, KANDK, LPRT, PETASET, PRESCR, PRESSIT, QUICK, SETUP, SOLA, THIRD, TILDE, VAPOR, VAPOR1, VAPOR3D,** and **VFCONV**
In common 'vof3dcom' /SSCM1/
Default = 0.02

EPSI (DOUBLE PRECISION) Pressure iteration convergence criterion.
Modified in **PRESCR** and **RINPUT**
Used in **PRESSIT** and **SOLA**
In common 'vof3dcom' /SSCM1/
Default = 0.001

FLHT (DOUBLE PRECISION) Initial fluid height in computing mesh (if appropriate).
Modified in **RINPUT** and **SETUP**
Used in
In common 'vof3dcom' /SSCM4/
Default = 1.0

GX (DOUBLE PRECISION) Body acceleration in the positive x-direction
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM1/
Default = 0.0

GY (DOUBLE PRECISION) Body acceleration in the positive y-direction
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM1/
Default = 0.0

GZ (DOUBLE PRECISION) Body acceleration in the positive z-direction
Modified in **RINPUT**
Used in **KANDK, PRESCR, QUICK, SETUP, THIRD,** and **TILDE**
In common 'vof3dcom' /SSCM1/
Default = 0.0

ICSURF (INTEGER) Flag for the initial fluid configuration generator.
ICSURF = 0 Horizontal or equilibrium surface.
ICSURF = 1 Axisymmetric free surface (see namelist FLUIDGN)
ICSURF = 2 Non-axisymmetric free surface (see namelist FLUIDGN)
Modified in **RINPUT**
Used in **SETFS** and **SETUP**
In common 'vof3dcom' /SSCM4A/
Default = 0

IDEFM (INTEGER) Defoamer flag
IDEFM = 0 Defoamer off
IDEFM = 1 Defoamer on
Modified in **RINPUT**
Used in **PRESCR** and **VFCONV**
In common 'vof3dcom' /SSCM1A/
Default = 0

IEQUIB (INTEGER) Equilibrium surface computation flag (requires ICSURF = 0 to be active).
IEQUIB = 0 Do not generate equilibrium surface.
IEQUIB = 1 Generate equilibrium surface.
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM4A/
Default = 0

IORDER (INTEGER) Flag for second order accurate option for the convective terms. (Does not seem to work in the code as received from Los Alamos).
Modified in **RINPUT**
Used in **SETUP** and **SOLA**
In common 'vof3dcom' /SSCM4A/
Default = 1

ISOR (INTEGER) Pressure equation solution flag
ISOR = 0 Use the Conjugate gradient method
ISOR = 1 Use Successive-Over-Relaxation
Modified in **RINPUT**
Used in **BC**, **BCFS**, **DELTADJ**, and **SOLA**
In common 'vof3dcom' /SSCM1A/
Default = 0

ISURFT (INTEGER) Surface tension computation flag
ISURFT = 0 No surface tension computations are performed
ISURFT = 1 Surface tension forces computed using either **SURF10N** (cylindrical coordinates) or **SURCART** (Cartesian coordinates)
Modified in **RINPUT**
Used in **PETACAL**
In common 'vof3dcom' /SSCM4A/
Default = 0

JNM (CHARACTER*8) Job identifier
Modified in **RINPUT**
Used in **LPRT**
In common 'vof3dcom' /SSCM1B/
Default = 'run 1'

LPR (INTEGER) Output flag
LPR = 0 No prints or plots
LPR = 1 Plots only
LPR = 2 Plots and prints
LPR = 3 Prints only
Modified in **RINPUT**
Used in **LPRT, MESHSET, SETFS,** and **SOLA**
In common 'vof3dcom' /SSCM1A/
Default = 2

NAME (CHARACTER*64) Job name
Modified in **RINPUT**
Used in **LPRT**
In common 'vof3dcom' /SSCM1B/
Default = 'prob. no name'

NFCAL (INTEGER) Flag to determine NF computation algorithm
NFCAL = 1 Use provisional value
NFCAL = 2 Use slope value
NFCAL = 3 Use decision-making algorithm to decide between provisional and slope values
Modified in **RINPUT**
Used in **SURCART** and **SURF10N**
In common 'vof3dcom' /SSCM2A/
Default = 3

NOWALL (INTEGER) Wall adhesion indicator
NOWALL = 0 Wall adhesion
NOWALL = 1 No wall adhesion
Modified in **RINPUT**
Used in **SURCART** and **SURF10N**
In common 'vof3dcom' /SSCM4A/
Default = 0

NU (DOUBLE PRECISION) Coefficient of kinematic viscosity
Modified in **RINPUT**
Used in **KANDK, QUICK, SETUP, THIRD, TILDE,** and **VISC3D**
In common 'vof3dcom' /SSCM2/
Default = 0.0

OMG (DOUBLE PRECISION) SOR acceleration parameter
Modified in **RINPUT**
Used in **BETACAL** and **PETASET**
In common 'vof3dcom' /SSCM2/
Default = 1.0

PLTDT (DOUBLE PRECISION) Time increment between data plots (if operative)
Modified in **RINPUT**
Used in **SETUP** and **SOLA**
In common 'vof3dcom' /SSCM2/
Default = 1.0

PRTDT (DOUBLE PRECISION) Time increment between data prints (if operative)
Modified in **RINPUT**
Used in **SETUP** and **SOLA**
In common 'vof3dcom' /SSCM2/
Default = 1.0

RADPS (DOUBLE PRECISION) Constant angular rotation velocity
Modified in **RINPUT**
Used in **KANDK**, **QUICK**, **THIRD**, and **TILDE**
In common 'vof3dcom' /SSCM5/
Default = 0.0

RHOF (DOUBLE PRECISION) Fluid Density
Modified in **RINPUT**
Used in
In common 'vof3dcom' /SSCM4/
Default = 1.0

SIGMA (DOUBLE PRECISION) Fluid surface tension divided by the fluid density
Modified in **RINPUT**
Used in **SETUP**, **SURCART**, and **SURF10N**
Default = 0.0

T (DOUBLE PRECISION) Problem time
Modified in **DELTADJ**, **RINPUT**, **SETUP**, and **SOLA**
Used in **DRAW**, **LPRT**, and **VFCONV**
In common 'vof3dcom' /SSCM2/
Default = 0.0

TD (INTEGER) Dump number for restart if greater than zero
Modified in **RINPUT**
Used in **SOLA**
In common 'vof3dcom' /SSCM2A/
Default = -1.0

TDDT (DOUBLE PRECISION) Time increment between restart tape dumps
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM2/
Default = 1.0e10

TLIMD (DOUBLE PRECISION) Problem run parameter (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /SSCM2/
Default = 1.0

TWFIN (DOUBLE PRECISION) Problem time to end calculation
Modified in **RINPUT**
Used in **SOLA**
In common 'vof3dcom' /SSCM2/
Default = 10.0

UI (DOUBLE PRECISION) Initial x-component of velocity for fluid cells
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM2/
Default = 0.0

VELMX (DOUBLE PRECISION) Maximum velocity expected (artifact from removed graphics routines)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /SSCM2/
Default = 2.0

VI (DOUBLE PRECISION) Initial y-component of velocity for fluid cells
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM2/
Default = 0.0

WB (INTEGER) Bottom wall boundary condition flag
WB = 1 Slip wall (symmetry plane)
WB = 2 No-slip wall with specified velocity
WB = 3 Continuative outlet boundary with all zero normal gradients
WB = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

WBK (INTEGER) Back wall boundary condition flag
WBK = 1 Slip wall (symmetry plane)
WBK = 2 No-slip wall with specified velocity
WBK = 3 Continuative outlet boundary with all zero normal gradients
WBK = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, PRESCR, SETFS, SURF10N,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

WF (INTEGER) Front wall boundary condition flag
WF = 1 Slip wall (symmetry plane)
WF = 2 No-slip wall with specified velocity
WF = 3 Continuative outlet boundary with all zero normal gradients
WF = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, SETFS, SURF10N, VAPOR, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

WI (DOUBLE PRECISION) Initial z-component of velocity for fluid cells
Modified in **RINPUT**
Used in **SETUP**
In common 'vof3dcom' /SSCM2/
Default = 0.0

WL (INTEGER) Left wall boundary condition flag
WL = 1 Slip wall (symmetry plane)
WL = 2 No-slip wall with specified velocity
WL = 3 Continuative outlet boundary with all zero normal gradients
WL = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, VAPOR, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

WR (INTEGER) Right wall boundary condition flag
WR = 1 Slip wall (symmetry plane)
WR = 2 No-slip wall with specified velocity
WR = 3 Continuative outlet boundary with all zero normal gradients
WR = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, VAPOR, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

WT (INTEGER) Top wall boundary condition flag
WT = 1 Slip wall (symmetry plane)
WT = 2 No-slip wall with specified velocity
WT = 3 Continuative outlet boundary with all zero normal gradients
WT = 4 Periodic boundary condition
Modified in **RINPUT**
Used in **ASET, BC, DRAW, PRESCR, VAPOR, VAPOR3D,** and **VISC3D**
In common 'vof3dcom' /SSCM2A/
Default = 1

Additional variables in XPUT added by John F. McKibben at IPST

DTCRMX (DOUBLE PRECISION) Arbitrary time step limit to ensure stability
Modified in **RINPUT**
Used in **DELTADJ**
In common 'vof3dcom' /MCKIB/
Default = 1.0e10

EPSIV (DOUBLE PRECISION) Vapor phase convergence criterion
Modified in **RINPUT**
Used in **VAPOR** and **VAPOR3D**
In common 'vof3dcom' /MCKIB/
Default = 0.00001

IFX(n) (INTEGER) I-coordinate location of the inside of a corner at static contact point (n)
Modified in **RINPUT**
Used in **VFCONV**
In common 'vof3dcom' /MCKIB1/
Default = 0

IORIN(m,n) Flags the orientation of the fluid flowing past a static contact point (see sample problem studying the die-swell problem)
IORIN(1,n) defines the side of the cell adjacent to the fluid
IORIN(2,n) defines the side of the cell adjacent to the vapor
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

ISLIP (INTEGER) Simple-minded attempt to treat a dynamic contact line
ISLIP = 0 No-slip at the contact line
ISLIP = 1 Allow slip in the cell at the contact line
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

ISLP(n) (INTEGER) Flag for slip along the wall leading up to the static contact point
ISLIP(n) = 0 No-slip
ISLIP(n) = 1 Slip
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

ISTRESS (INTEGER) Deviatoric normal stress flag
ISTRESS = 0 Do not compute
ISTRESS = 1 Compute
Modified in **RINPUT**
Used in **VISC3D**
In common 'vof3dcom' /MCKIB1/
Default = 0

IVFR(n) (INTEGER) I-coordinate of the reference cell for vapor region (n)
Modified in **RINPUT**
Used in **VAPOR** and **VAPOR3D**
In common 'vof3dcom' /MCKIB1/
Default = 2

IVWB (INTEGER) Bottom face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

IVWF (INTEGER) Front face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

IVWK (INTEGER) Back face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

IVWL (INTEGER) Left face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

IVWR (INTEGER) Right face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

IVWT (INTEGER) Top face vapor phase boundary flag (currently not implemented)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

JVFR(n) (INTEGER) J-coordinate of the reference cell for vapor region (n)
Modified in **RINPUT**
Used in **VAPOR3D**
In common 'vof3dcom' /MCKIB1/
Default = 2

KFX(n) (INTEGER) K-coordinate location of the inside of a corner at static contact point (n)
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

KVFR(n) (INTEGER) K-coordinate of the reference cell for vapor region (n)
Modified in **RINPUT**
Used in **VAPOR** and **VAPOR3D**
In common 'vof3dcom' /MCKIB1/
Default = 2

LVAPOR (INTEGER) Flag to solve the vapor phase regions
LVAPOR = 0 Do not solve void regions
LVAPOR = 1 Solve regions using previous value as initial guess
LVAPOR = 2 Solve regions using 0 as initial guess at all times
Modified in **RINPUT**
Used in **SOLA**, **VAPOR**, and **VAPOR3D**
In common 'vof3dcom' /MCKIB1/
Default = 0

LVFLAG (INTEGER) Flag to label the void regions
LVFLAG = 0 Do not label
LVFLAG = 1 Label
Modified in **RINPUT**
Used in **LAVORE**
In common 'vof3dcom' /MCKIB1/
Default = 0

NFX (INTEGER) Number of static contact lines present
Modified in **RINPUT**
Used in
In common 'vof3dcom' /MCKIB1/
Default = 0

NVFR (INTEGER) Number of vapor phase regions to be studied
Modified in **RINPUT**
Used in **VAPOR** and **VAPOR3D**
In common 'vof3dcom' /MCKIB1/
Default = 0

RHOG (DOUBLE PRECISION) Ration of the vapor phase density to the liquid phase density
Modified in **RINPUT**
Used in **PRESCR** and **VAPOR1**
In common 'vof3dcom' /MCKIB/
Default = 0.0

UBW (DOUBLE PRECISION) U-velocity component along the bottom wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

UFW (DOUBLE PRECISION) U-velocity component along the front wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

UKW (DOUBLE PRECISION) U-velocity component along the back wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

UTW (DOUBLE PRECISION) U-velocity component along the top wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

VBW (DOUBLE PRECISION) V-velocity component along the bottom wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

VLW (DOUBLE PRECISION) V-velocity component along the left wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

VOMG (DOUBLE PRECISION) SOR acceleration parameter for the vapor phase solution
Modified in **RINPUT**
Used in **VAPOR** and **VAPOR3D**
In common 'vof3dcom' /MCKIB/
Default = 1.0

VRW (DOUBLE PRECISION) V-velocity component along the right wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

VTW (DOUBLE PRECISION) V-velocity component along the top wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

WFW (DOUBLE PRECISION) W-velocity component along the front wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

WKW (DOUBLE PRECISION) W-velocity component along the back wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

WLW (DOUBLE PRECISION) W-velocity component along the left wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

WRW (DOUBLE PRECISION) W-velocity component along the right wall (no-slip condition)
Modified in **RINPUT**
Used in **BC**
In common 'vof3dcom' /MCKIB/
Default = 0.0

Variables in namelist MESHGN read in **MESHSET**

Variables in MESHGN pertaining to the grid structure

DXMN(n) (DOUBLE PRECISION) Min. cell spacing adjacent to "focal" point in the x-direction in region (n)
Modified in **MESHX**
Used in **MESHSET**
In common 'vof3dcom' /SSCM1/

DYMN(n) (DOUBLE PRECISION) Min. cell spacing adjacent to "focal" point in the y-direction in region (n)
Modified in **MESHY**
Used in **MESHSET**
In common 'vof3dcom' /SSCM1/

DZMN(n) (DOUBLE PRECISION) Min. cell spacing adjacent to "focal" point in the z-direction in region (n)
Modified in **MESHZ**
Used in **MESHSET**
In common 'vof3dcom' /SSCM1/

NKX (INTEGER) Number of regions in the x-direction
Modified in **MESHSET**
Used in **MESHX**
In common none

NKY (INTEGER) Number of regions in the y-direction
Modified in **MESHSET**
Used in **MESHY**
In common none

NKZ (INTEGER) Number of regions in the z-direction
Modified in **MESHSET**
Used in **MESHZ**
In common none

NXL(n) (INTEGER) Number of cells to the left of the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHX**
In common 'vof3dcom' /SSCM2A/

NXR(n) (INTEGER) Number of cells to the right of the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHX**
In common 'vof3dcom' /SSCM2A/

NYL(n) (INTEGER) Number of cells in front of the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHY**
In common 'vof3dcom' /SSCM2A/

NYR(n) (INTEGER) Number of cells in back of the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHY**
In common 'vof3dcom' /SSCM2A/

NZL(n) (INTEGER) Number of cells below the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHZ**
In common 'vof3dcom' /SSCM2A/

NZR(n) (INTEGER) Number of cells above the "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHZ**
In common 'vof3dcom' /SSCM2A/

XC(n) (DOUBLE PRECISION) Location of the x-direction "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHX**
In common 'vof3dcom' /SSCM2/

XL(n+1) (DOUBLE PRECISION) Location of the left boundary of region (n) [the right boundary is (n+1)]
Modified in
Used in **MESHSET** and **MESHX**
In common 'vof3dcom' /SSCM2/

YC(n) (DOUBLE PRECISION) Location of the y-direction "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHY**
In common 'vof3dcom' /SSCM2/

YL(n+1) (DOUBLE PRECISION) Location of the front boundary of region (n) [the back boundary is (n+1)]
Modified in
Used in **MESHSET** and **MESHY**
In common 'vof3dcom' /SSCM2/

ZC(n) (DOUBLE PRECISION) Location of the z-direction "focal" point in region (n)
Modified in
Used in **MESHSET** and **MESHZ**
In common 'vof3dcom' /SSCM2/

ZL(n+1) (DOUBLE PRECISION) Location of the bottom boundary of region (n) [the top boundary is (n+1)]
Modified in
Used in **MESHSET** and **MESHZ**
In common 'vof3dcom' /SSCM2/

Variables in MESHGN pertaining to the interior obstacles

IOH(n) (INTEGER) Obstacle flag for obstacle (n)
IOH(n) = 0 Subtract obstacles within region
IOH(n) = 1 Add obstacles within region
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5B/
Default = 0

NOBS (INTEGER) Number of interior obstacles
Modified in **MESHSET**
Used in **ASET**
In common 'vof3dcom' /SSCM5B/
Default = 0

OA1(n) (DOUBLE PRECISION) Coefficient of x-term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

OA2(n) (DOUBLE PRECISION) Coefficient of x^2-term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

OB1(n) (DOUBLE PRECISION) Coefficient of z-term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

OB2(n) (DOUBLE PRECISION) Coefficient of z^2-term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

OC1(n) (DOUBLE PRECISION) Coefficient of constant term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

OC2(n) (DOUBLE PRECISION) Coefficient of xz-term in function
Modified in
Used in **ASET** and **MESHSET**
In common 'vof3dcom' /SSCM5/
Default = 0.0

Variables in Namelist FLUIDGN read in **SETFS**

IQH(n) (INTEGER) Flag for adding fluid within a region
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5B/
Default = 0

NQBS (INTEGER) Number of fluid generation regions
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5B/
Default = 0

Axisymmetric (ICSURF = 1)

QA1(n) (DOUBLE PRECISION) Coefficient of x-term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QA2(n) (DOUBLE PRECISION) Coefficient of x^2-term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QB1(n) (DOUBLE PRECISION) Coefficient of z-term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QB2(n) (DOUBLE PRECISION) Coefficient of z^2-term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QC1(n) (DOUBLE PRECISION) Coefficient of constant term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QC2(n) (DOUBLE PRECISION) Coefficient of xz-term in function
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

Non-axisymmetric (ICSURF>1)

QA1(n) (DOUBLE PRECISION) Value of X at the center of volume n
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QA2(n) (DOUBLE PRECISION) Lower limit of I index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QA2(n) (DOUBLE PRECISION) Upper limit of I index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QB1(n) (DOUBLE PRECISION) Value of Y at the center of volume n
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QB2(n) (DOUBLE PRECISION) Lower limit of J index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QB2(n) (DOUBLE PRECISION) Upper limit of J index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QC1(n) (DOUBLE PRECISION) Value of Z at the center of volume n
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QC2(n) (DOUBLE PRECISION) Lower limit of K index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QC2(n) (DOUBLE PRECISION) Upper limit of K index
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QD1(n) (DOUBLE PRECISION) Constant term in function n
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QD2(n) (DOUBLE PRECISION) Indicator specifying if the fluid is initially fluid or void
QD2(n) = 0.0 Void
QD2(n) = 1.0 Fluid
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

QD3(n) (DOUBLE PRECISION) Not currently used
Modified in **SETFS**
Used in
In common 'vof3dcom' /SSCM5/
Default = 0.0

APPENDIX II

VARIABLE GRID QUICK DIFFERENCING

The initial literature derivations for Quadratic Upstream Interpolation for Convective Kinematics (QUICK) differencing were formulated for constant grid problems. Since, for computational efficiency, it is necessary to use variable grid spacing, a variable grid representation of the QUICK algorithm was necessary.

QUICK differencing was first proposed by Leonard[85,112] as a technique for improving the accuracy and stability of the finite difference representation of the convective terms in the NSE. The basic premise is that more accurate differencing is possible if the flux at each cell face is computed from a quadratic interpolation of the local velocities. For added stability, the interpolation is weighted in the upstream direction. Figure II-1 shows the local grid configuration for a two dimensional problem.

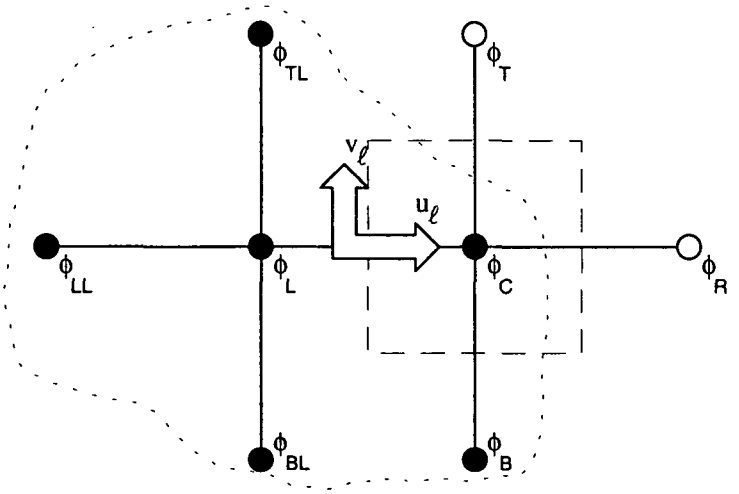

Figure II-1. Schematic of QUICK interpolation.

From Leonard's derivation, the two dimensional formula at the left cell face is

$$\phi_\ell = \phi_{LIN} - \frac{1}{8}\text{CURVN} + \frac{1}{24}\text{CURVT}, \tag{II-1}$$

where

$$\phi_{LIN} = \frac{1}{2}(\phi_L + \phi_C) \tag{II-2}$$

is a linear interpolation term which alone would yield second order accurate central differencing. The term, CURVN, defined as

$$\begin{aligned} &\text{CURVN} = \phi_C - 2\phi_L + \phi_{LL} && (u_\ell > 0) \\ \text{and}\quad &\text{CURVN} = \phi_R - 2\phi_C + \phi_L && (u_\ell < 0), \end{aligned} \tag{II-3}$$

represents the upstream difference normal to the cell face. Finally, CURVT, defined as

$$\begin{aligned} &\text{CURVT} = \phi_{TL} - 2\phi_L + \phi_{BL} && (u_\ell > 0) \\ \text{and}\quad &\text{CURVT} = \phi_T - 2\phi_C + \phi_B && (u_\ell < 0), \end{aligned} \tag{II-4}$$

represents the small tangential component of the curvature. Note that the point ϕ_B cancels out in Leonard's[85] derivation, and thus does not appear in the formulas for $u_\ell > 0$. In addition, it has been found through experience that the small CURVT terms may be neglected without a major impact on the resulting accuracy.[113]

If I substitute the terms, with the exception of CURVT, into (II-1), the resulting formulas are

$$\begin{aligned} &\phi_\ell = \frac{1}{8}(3\phi_C + 6\phi_L - \phi_{LL}) && (u_\ell > 0) \\ \text{and}\quad &\phi_\ell = \frac{1}{8}(3\phi_L + 6\phi_C - \phi_R) && (u_\ell < 0). \end{aligned} \tag{II-5}$$

Combining the results from Equation (II-5) with the analogous interpolation formulas at the right cell face yields the following finite difference formulas for $\partial\phi/\partial x$:

$$\frac{\partial\phi}{\partial x} = \frac{1}{8\Delta x}\left(3\phi_R + 3\phi_C - 7\phi_L + \phi_{LL}\right) \qquad (u_C > 0)$$

and $$\frac{\partial\phi}{\partial x} = \frac{1}{8\Delta x}\left(-\phi_{RR} + 7\phi_R - 3\phi_C - 3\phi_L\right) \qquad (u_C < 0). \qquad \text{(II-6)}$$

Equation (II-6) represents the QUICK formulas for a constant grid when the CURVT terms are neglected. For a variable grid, similar formulas may be defined. The simplest method for deriving these formulas is to fit a quadratic interpolation formula to the data points surrounding the appropriate cell face in the appropriate direction. Then compute the difference as outlined above. The geometry used in the derivation is presented in Figure II-2.

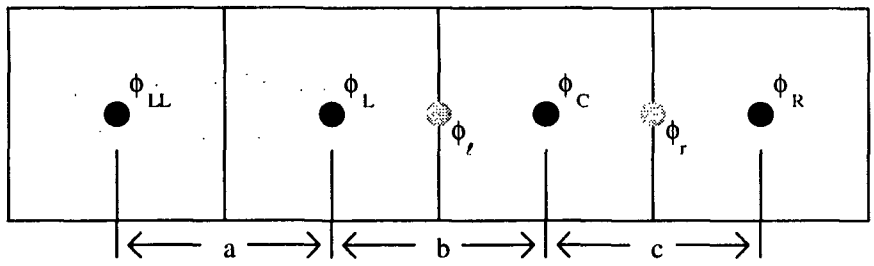

Figure II-2. Cell Spacing Schematic.

We begin by fitting a quadratic interpolation formula of the form $\phi = r + sx + tx^2$ to the neighboring points. Then, after substituting the location of the cell face into the quadratic interpolation formula, it is possible to determine the correct weighting factors. In matrix form, the equations solved for the interpolation are

$$\begin{bmatrix} 1 & -a & a^2 \\ 1 & 0 & 0 \\ 1 & b & b^2 \end{bmatrix} \begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} \phi_{LL} \\ \phi_L \\ \phi_C \end{bmatrix} \qquad \text{(II-7)}$$

with the reference location at ϕ_L. Rearranging the equations yields

$$\begin{bmatrix} -a & a^2 & 1 \\ b & b^2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ r \end{bmatrix} = \begin{bmatrix} \phi_{LL} \\ \phi_C \\ \phi_L \end{bmatrix}. \tag{II-8}$$

which simplifies to

$$\begin{bmatrix} -1 & a & 1/a \\ 0 & a+b & (a+b)/ab \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ r \end{bmatrix} = \begin{bmatrix} \phi_{LL}/a \\ (b\phi_{LL} + a\phi_C)/ab \\ \phi_L \end{bmatrix} \tag{II-9}$$

Thus,

$$r = \phi_L, \tag{II-10}$$

$$\begin{aligned} t &= \left[(b\phi_{LL} + a\phi_C)/ab - (a+b)\phi_L/ab\right]/(a+b) \\ &= \left[a(\phi_C - \phi_L) + b(\phi_{LL} - \phi_L)\right]/\left[ab(a+b)\right], \end{aligned} \tag{II-11}$$

and

$$\begin{aligned} s &= -\left(\phi_{LL}/a - \phi_L/a - a\left[a(\phi_C - \phi_L) + b(\phi_{LL} - \phi_L)\right]/\left[ab(a+b)\right]\right) \\ &= \left[a^2(\phi_C - \phi_L) + ab(\phi_{LL} - \phi_L) - b(a+b)(\phi_{LL} - \phi_L)\right]/\left[ab(a+b)\right] \\ &= \left[a^2(\phi_C - \phi_L) - b^2(\phi_{LL} - \phi_L)\right]/\left[ab(a+b)\right]. \end{aligned} \tag{II-12}$$

Substituting the results of Equations (II-10), (II-11), and (II-12) and the position of the cell face, b/2, into the interpolation formula yields

$$\phi_\ell = \phi_L + \frac{a^2(\phi_C - \phi_L) - b^2(\phi_{LL} - \phi_L)}{ab(a+b)}\frac{b}{2} + \frac{a(\phi_C - \phi_L) + b(\phi_{LL} - \phi_L)}{ab(a+b)}\frac{b^2}{4}, \tag{II-13}$$

which simplifies, in agreement with results from the literature,[113] to

$$\phi_\ell = \frac{-b^2}{4a(a+b)}\phi_{LL} + \frac{2a+b}{4a}\phi_L + \frac{2a+b}{4(a+b)}\phi_C, \tag{II-14}$$

with an analogous result for the other cell face

$$\phi_r = \frac{-c^2}{4b(b+c)}\phi_L + \frac{2b+c}{4b}\phi_C + \frac{2b+c}{4(b+c)}\phi_R. \tag{II-15}$$

When a and b are equal, (II-14) reduces to the constant grid formula, (II-5), as expected.

The complete finite difference formula for the variable grid representation of QUICK differencing is derived by talking a central difference about ϕ_C

$$\frac{\partial \phi}{\partial x} = \frac{\phi_r - \phi_l}{(b+c)/2}$$

$$= \frac{1}{(b+c)/2}\left\{\frac{-b^2}{4a(a+b)}\phi_{LL} - \left[\frac{2a+b}{4a} + \frac{c^2}{4b(b+c)}\right]\phi_L \right. \qquad \text{(II-16)}$$

$$\left. + \left[\frac{2b+c}{4b} + \frac{2a+b}{4(a+b)}\right]\phi_C + \frac{2b+c}{4(b+c)}\phi_R \right\} \qquad u_C > 0.$$

Similar formulas were derived for the other flow directions.

APPENDIX III

VARIABLE GRID THIRD ORDER ACCURATE UPWIND DIFFERENCING

The third order accurate upwind differencing scheme was first proposed by Agarwal[86] as a method for increased accuracy and numerical stability at high speeds. Agarwal's derivation of a constant grid formula will be presented followed by an alternative derivation based on the method of undetermined coefficients[114] which is then extended to a variable grid formulation. Computational grid spacings will be with reference to Figure III-1.

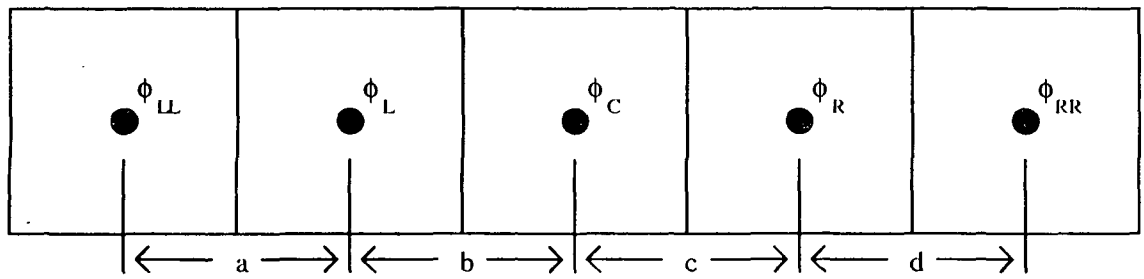

Figure III-1 Grid spacings for derivation of third order accurate upwind differencing.

Agarwal begins with the second order central difference formula at the point C which can be written as

$$\left.\frac{\partial\phi}{\partial x}\right|_C = \frac{\phi_R - \phi_L}{2\Delta x} - \frac{\Delta x^2}{6}\left.\frac{\partial^3\phi}{\partial x^3}\right|_C + \cdots. \qquad \text{(III-1)}$$

Now if $u > 0$, then

$$\left.\frac{\partial^3\phi}{\partial x^3}\right|_C = \left[\left.\frac{\partial^2\phi}{\partial x^2}\right|_C - \left.\frac{\partial^2\phi}{\partial x^2}\right|_L\right] \Big/ \Delta x$$
$$= \frac{(\phi_R - 2\phi_C + \phi_L) - (\phi_C - 2\phi_L + \phi_{LL})}{\Delta x^3} \qquad \text{(III-2)}$$
$$= \frac{\phi_R - 3\phi_C + 3\phi_L - \phi_{LL}}{\Delta x^3}.$$

Similarly, if $u < 0$, then

$$\left.\frac{\partial^3\phi}{\partial x^3}\right|_C = \left[\left.\frac{\partial^2\phi}{\partial x^2}\right|_R - \left.\frac{\partial^2\phi}{\partial x^2}\right|_C\right] \Big/ \Delta x$$
$$= \frac{(\phi_{RR} - 2\phi_R + \phi_C) - (\phi_R - 2\phi_C + \phi_L)}{\Delta x^3} \qquad \text{(III-3)}$$
$$= \frac{\phi_{RR} - 3\phi_R + 3\phi_C - \phi_L}{\Delta x^3}.$$

Substituting (III-2) and (III-3) into (III-1) yields Agarwal's third order accurate upwind technique for constant grid spacings:

$$\frac{\partial\phi}{\partial x} = \frac{1}{6\Delta x}(2\phi_R + 3\phi_C - 6\phi_L + \phi_{LL}) \qquad (u > 0)$$
$$\text{and} \quad \frac{\partial\phi}{\partial x} = \frac{1}{6\Delta x}(-\phi_{RR} + 6\phi_R - 3\phi_C - 2\phi_L) \qquad (u < 0). \qquad \text{(III-4)}$$

Since I am interested in problems using variable grid spacings, I used an alternative derivation technique that is easily extended to variable grid spacings. This derivation, using the method of undetermined coefficients,[114] follows.

First, I define the derivative of interest in terms of several base points and coefficients,

$$\frac{\partial\phi}{\partial x} = c_{LL}\phi_{LL} + c_L\phi_L + c_C\phi_C + c_R\phi_C. \qquad \text{(III-5)}$$

Next I want to determine a polynomial that passes through these points exactly (in this case of degree 3). Thus, I define four equations for the polynomial:

$$\left.\frac{\partial x^3}{\partial x}\right|_C = c_{LL}(x_C - 2\Delta x)^3 + c_L(x_C - \Delta x)^3 + c_C(x_C)^3 + c_R(x_C + \Delta x)^3, \qquad \text{(III-6a)}$$

$$\left.\frac{\partial x^2}{\partial x}\right|_C = c_{LL}(x_C - 2\Delta x)^2 + c_L(x_C - \Delta x)^2 + c_C(x_C)^2 + c_R(x_C + \Delta x)^2, \qquad \text{(III-6b)}$$

$$\left.\frac{\partial x^1}{\partial x}\right|_C = c_{LL}(x_C - 2\Delta x)^1 + c_L(x_C - \Delta x)^1 + c_C(x_C)^1 + c_R(x_C + \Delta x)^1, \qquad \text{(III-6c)}$$

and

$$\left.\frac{\partial x^0}{\partial x}\right|_C = c_{LL}(x_C - 2\Delta x)^0 + c_L(x_C - \Delta x)^0 + c_C(x_C)^0 + c_R(x_C + \Delta x)^0. \qquad \text{(III-6d)}$$

Since the location of x_C is arbitrary, it is set equal to zero yielding the following system of equations in matrix form,

$$\begin{bmatrix} -8 & -1 & 0 & 1 \\ 4 & 1 & 0 & 1 \\ -2 & -1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_{LL} \\ c_L \\ c_C \\ c_R \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1/\Delta x \\ 0 \end{bmatrix}. \qquad \text{(III-7)}$$

When this system of equations is solved, it yields the same formula as that derived by Agarwal[86] for $u > 0$. A similar procedure can be followed to derive the formula for $u < 0$.

When a variable grid is used, the same procedure is used, but the system of equations becomes more complex:

$$\begin{bmatrix} -(a+b)^3 & -b^3 & 0 & c^3 \\ (a+b)^2 & b^2 & 0 & c^2 \\ -(a+b) & -b & 0 & c \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_{LL} \\ c_L \\ c_C \\ c_R \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \qquad \text{(III-8)}$$

Solution of this system of equations yields the variable grid finite difference formulas

$$\frac{\partial \phi}{\partial x} = \frac{bc}{a(a+b)(a+b+c)}\phi_{LL} - \frac{c(a+b)}{ab(b+c)}\phi_L + \frac{bc-(b-c)(a+b)}{bc(a+b)}\phi_C + \frac{b(a+b)}{c(b+c)(a+b+c)}\phi_R \qquad (u > 0) \qquad \text{(III-9a)}$$

$$\frac{\partial \phi}{\partial x} = -\frac{c(c+d)}{b(b+c)(b+c+d)}\phi_L - \frac{bc+(b-c)(c+d)}{bc(c+d)}\phi_C$$

and

$$+\frac{b(c+d)}{cd(b+c)}\phi_R - \frac{bc}{d(c+d)(b+c+d)}\phi_{RR} \qquad (u<0). \qquad \text{(III-9b)}$$

These finite difference formulas and analogous formulas for the remaining velocity components yield the third order accurate upwind differencing scheme. As indicated above, this method was proposed by Agarwal[86]. I have extended the algorithm to allow problems with variable grids and to be studied. Within the IPST-VOF3D program, this technique is implemented in the subroutine THIRD.

APPENDIX IV

VARIABLE GRID KAWAMURA AND KUWAHARA METHOD

I will begin deriving an approximate version of Kawamura and Kuwahara's[87] method for treating the convective terms in the NSE by reviewing the derivation of the constant grid formula. This is followed by an attempt to directly reproduce their derivation scheme for a variable grid. Finally, I present my approximation of their method for a variable grid. The notation presented in Figure IV-1 will be used with the constant grid spacing denoted Δx.

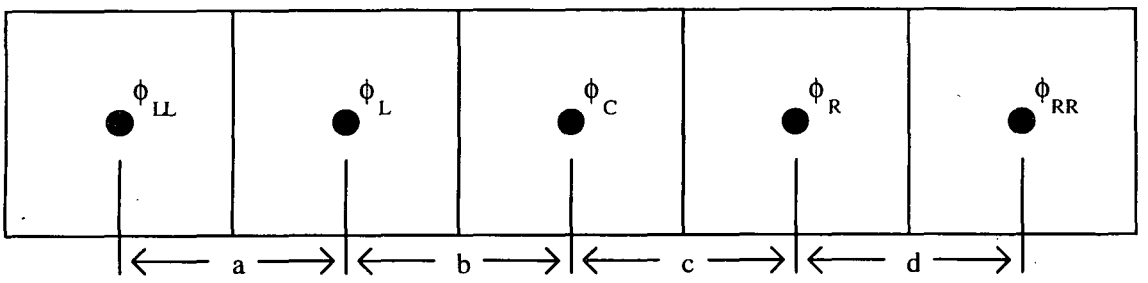

Figure IV-1. Grid spacings for derivation of Kawamura and Kuwahara's technique.

KAWAMURA AND KUWAHARA'S DERIVATION

We begin with a second order upwind differencing scheme,

$$\left.\frac{\partial\phi}{\partial x}\right|_C = \frac{1}{2h}\left(3\phi_C - 4\phi_L + \phi_{LL}\right) \qquad u_C > 0 \qquad \text{(IV-1a)}$$

and

$$\left.\frac{\partial\phi}{\partial x}\right|_C = \frac{1}{2h}\left(-\phi_{RR} + 4\phi_R - 3\phi_C\right) \qquad u_C < 0. \qquad \text{(IV-1b)}$$

These formulas can be combined to yield a single formula independent of the flow direction

$$\left(u_C \frac{\partial \phi}{\partial x}\right)_C = \frac{u_C}{4\Delta x}(-\phi_{RR} + 4\phi_R - 4\phi_L + \phi_{LL}) + \frac{|u_C|}{4\Delta x}(\phi_{RR} - 4\phi_R + 6\phi_C - 4\phi_L + \phi_{LL}). \tag{IV-2}$$

From Taylor series expansions, this formula can be rewritten as

$$\left(u \frac{\partial \phi}{\partial x}\right)_C = u_C\left[\frac{\partial \phi}{\partial x} - \frac{\Delta x^2}{2}\frac{\partial^3 \phi}{\partial x^3} + O(\Delta x^4)\right] + |u_C|\left[\Delta x^3 \frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^5)\right]. \tag{IV-3}$$

Thus, the leading error in (IV-1) or (IV-2) can be reduced by eliminating the term

$$\frac{\Delta x^2}{2}\frac{\partial^3 \phi}{\partial x^3}. \tag{IV-4}$$

Improved accuracy is obtained by replacing the first term in (IV-3),

$$\frac{\partial \phi}{\partial x} - \frac{\Delta x^2}{2}\frac{\partial^3 u}{\partial x^3} + O(\Delta x^4), \tag{IV-5}$$

with

$$\frac{\partial \phi}{\partial x} + O(\Delta x^4) = \frac{-\phi_{RR} + 8\phi_R - 8\phi_L + \phi_{LL}}{12\Delta x}, \tag{IV-6}$$

yielding

$$\left(u \frac{\partial u}{\partial x}\right)_C = u_C\left[\frac{\partial \phi}{\partial x} + O(\Delta x^4)\right] + |u_C|\left[\Delta x^3 \frac{\partial^4 \phi}{\partial x^4} + O(\Delta x^5)\right]. \tag{IV-7}$$

The resulting analog of (IV-2) with an error of $O(\Delta x^4)$is

$$\left(u \frac{\partial \phi}{\partial x}\right)_C = \frac{u_C}{12\Delta x}(-\phi_{RR} + 8\phi_R - 8\phi_L + \phi_{LL}) + \frac{|u_C|}{4\Delta x}(\phi_{RR} - 4\phi_R + 6\phi_C - 4\phi_L + \phi_{LL}). \tag{IV-8}$$

ATTEMPTED DERIVATION WITH VARIABLE GRID

In this section, we follow the steps of Kawamura and Kuwahara's derivation as far as possible for a variable grid. All derivations will be with respect to the dimensions shown in Figure IV-1. The finite difference formulas used below were derived using the method of undetermined coefficients described by Gerald[114] and in Appendix II of this document.

For a variable grid, (IV-1) becomes

$$\left(u\frac{\partial\phi}{\partial x}\right)_C = u_C\left(\frac{a+2b}{b(a+b)}\phi_C - \frac{a+b}{ab}\phi_L + \frac{b}{a(a+b)}\phi_{LL}\right) \qquad u_C > 0 \qquad \text{(IV-9a)}$$

and

$$\left(u\frac{\partial\phi}{\partial x}\right)_C = u_C\left(-\frac{c}{d(c+d)}\phi_{RR} + \frac{c+d}{cd}\phi_R - \frac{2c+d}{c(c+d)}\phi_C\right) \qquad u_C < 0. \qquad \text{(IV-9b)}$$

Thus, (IV-2) can be rewritten as

$$\begin{aligned}\left(u\frac{\partial\phi}{\partial x}\right)_C = \frac{u_C}{2}\Bigg[&-\frac{c}{d(c+d)}\phi_{RR} + \frac{c+d}{cd}\phi_R + \left(-\frac{2c+d}{c(c+d)} + \frac{a+2b}{b(a+b)}\right)\phi_C \\ &-\frac{a+b}{ab}\phi_L + \frac{b}{a(a+b)}\phi_{LL}\Bigg] + \frac{|u_C|}{2}\Bigg[\frac{c}{d(c+d)}\phi_{RR} - \frac{c+d}{cd}\phi_R \\ &+\left(\frac{2c+d}{c(c+d)} + \frac{a+2b}{b(a+b)}\right)\phi_C - \frac{a+b}{ab}\phi_L + \frac{b}{a(a+b)}\phi_{LL}\Bigg].\end{aligned} \qquad \text{(IV-10)}$$

In order to continue with Kawamura and Kuwahara's derivation, we begin by defining the terms in (IV-3)

$$\frac{\partial\phi}{\partial x} + O(\Delta x^2) = \frac{b}{c(b+c)}\phi_R + \frac{c-b}{bc}\phi_C - \frac{c}{b(b+c)}\phi_L, \qquad \text{(IV-11)}$$

$$\frac{\partial^3\phi}{\partial x^3}+O(\Delta x^4)=\frac{6(a+2b-c)}{d(c+d)(b+c+d)(a+b+c+d)}(\phi_{RR}-\phi_C)$$
$$-\frac{6(a+2b-c-d)}{cd(a+b+c)(b+c)}(\phi_R-\phi_C)+\frac{6(d+2c-b-a)}{ab(b+c+d)(b+c)}(\phi_L-\phi_C) \quad \text{(IV-12)}$$
$$-\frac{6(d+2c-b)}{a(a+b)(a+b+c)(a+b+c+d)}(\phi_{LL}-\phi_C),$$

and
$$\frac{\partial^4\phi}{\partial x^4}+O(\Delta x^5)=\frac{6}{d(c+d)(b+c+d)(a+b+c+d)}(\phi_{RR}-\phi_C)$$
$$-\frac{6}{cd(a+b+c)(b+c)}(\phi_R-\phi_C)-\frac{6}{ab(b+c+d)(b+c)}(\phi_L-\phi_C) \quad \text{(IV-13)}$$
$$+\frac{6}{a(a+b)(a+b+c)(a+b+c+d)}(\phi_{LL}-\phi_C).$$

When (IV-11), (IV-12), and (IV-13) are substituted into (IV-3), it does not yield (IV-2), implying that the derivation of Kawamura and Kuwahara's method is not possible for variable grids.

APPROXIMATE DERIVATION WITH VARIABLE GRID

An approximate form of Kawamura and Kuwahara's technique can be derived by beginning the variable grid portion of the derivation with Equation (IV-7). Substituting the variable grid analogue of (IV-6),

$$\frac{\partial\phi}{\partial x}+O\left(\Delta x^4\right)=-\frac{bc(a+b)}{d(c+d)(b+c+d)(a+b+c+d)}(\phi_{RR}-\phi_C)$$
$$+\frac{b(a+b)(c+d)}{cd(a+b+c)(b+c)}(\phi_R-\phi_C)-\frac{c(a+b))(c+d)}{ab(b+c+d)(b+c)}(\phi_L-\phi_C) \quad \text{(IV-14)}$$
$$+\frac{bc(c+d)}{a(a+b)(a+b+c)(a+b+c+d)}(\phi_{LL}-\phi_C).$$

and (IV-13) into Equation (IV-7) results in an approximate formula for Kawamura and Kuwahara's method on a variable grid:

$$\left(u\frac{\partial\phi}{\partial x}\right)_C=\frac{-bc(a+b)\,\phi_C+24|\phi_C|}{d(c+d)(b+c+d)(a+b+c+d)}(\phi_{RR}-\phi_C)$$
$$+\frac{b(a+b)(c+d)\,\phi_C-24|\phi_C|}{cd(a+b+c)(b+c)}(\phi_R-\phi_C)$$
$$-\frac{c(a+b))(c+d)\,\phi_C-24|\phi_C|}{ab(b+c+d)(b+c)}(\phi_L-\phi_C) \quad \text{(IV-15)}$$
$$+\frac{bc(c+d)\,\phi_C+24|\phi_C|}{a(a+b)(a+b+c)(a+b+c+d)}(\phi_{LL}-\phi_C).$$

APPENDIX V

ALTERNATIVE FORMULATION FOR THE INTERFACIAL DEVIATORIC STRESS

The deviatoric normal stress at the interface is needed to accurately impose the interfacial boundary condition arising from the normal stress balance. Here I present a three-dimensional analogue of the method used by Hill[40,41] for determining the deviatoric normal stress. First I present Hill's derivation followed by my extension to three-dimensions.

TWO-DIMENSIONAL FORMULAS

I begin by defining the components of the stress tensor in two dimensions. I will use the subscripts to denote the direction.

$$\tau_{xx} = p - 2\mu e_{xx} = p - 2\mu\left(\frac{\partial u}{\partial x}\right) \tag{V-1a}$$

$$\tau_{yy} = p - 2\mu e_{yy} = p - 2\mu\left(\frac{\partial v}{\partial y}\right) \tag{V-1b}$$

$$\tau_{xy} = \tau_{yx} = -2\mu e_{xy} = -\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{V-1c}$$

The normal stress at the interface is computed from

$$NS = \mathbf{n}\cdot\tau\cdot\mathbf{n} = \tau_{xx}n_x^2 + \tau_{yy}n_y^2 + 2\tau_{xy}n_x n_y \tag{V-2}$$

and the tangential stress is computed from

$$TS = \mathbf{n}\cdot\tau\cdot\mathbf{t} = \tau_{xx}n_x t_x + \tau_{yy}n_y t_y + \tau_{xy}\left(n_x t_y + n_y t_x\right). \tag{V-3}$$

Finally, I will need the continuity equation

$$0 = e_{xx} + e_{yy} \tag{V-4}$$

Since we know that the tangential stress is zero at the interface (because the surface

tension is constant and the vapor phase viscosity is zero), we can write three equations for the normal stress,

$$NS = p - 2\mu e_{xx} n_x^2 - 2\mu e_{yy} n_y^2 + -4\mu e_{xy} n_x n_y; \tag{V-5}$$

the tangential stress,

$$0 = p(n_x t_x + n_y t_y) - 2\mu e_{xx} n_x t_x - 2\mu e_{yy} n_y t_y - 2\mu e_{xy}(n_x t_y + n_y t_x); \tag{V-6}$$

and the continuity equation,

$$0 = e_{xx} + e_{yy}. \tag{V-7}$$

Rearranging and making use of the orthogonality of the normal and tangential vectors yields the system of equations

$$\begin{bmatrix} n_x^2 & n_y^2 & 2n_x n_y \\ n_x t_x & n_y t_y & n_x t_y + n_y t_x \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ e_{xy} \end{bmatrix} = \begin{bmatrix} visc \\ 0 \\ 0 \end{bmatrix} \tag{V-8}$$

where $visc = (p - NS)/2\mu$.

Next, we solve this system of equations yielding

$$\begin{aligned} visc &= \left[(n_x^2 - n_y^2) - 2n_x n_y \frac{n_x t_x - n_y t_y}{n_x t_y + n_y t_x} \right] e_{xx} \\ &= \left[(n_y^2 - n_x^2) - 2n_x n_y \frac{n_y t_y - n_x t_x}{n_x t_y + n_y t_x} \right] e_{yy}. \end{aligned} \tag{V-9}$$

The appropriate formula from (V-9) is chosen depending on which coordinate direction is nearest the surface normal.

The final piece of information needed to complete the definition of the normal stress in two-dimension is the relation between the unit normal vector and the unit tangential vector. Since they are both unit vectors and must be orthogonal, $t_x = -n_y$ and $t_y = n_x$. Substituting these relations into (V-9) yields

$$\text{visc} = \left[\left(n_x^2 - n_y^2\right) + \frac{4n_x^2 n_y^2}{n_x^2 - n_y^2}\right] e_{xx} = -\left[\left(n_x^2 - n_y^2\right) + \frac{4n_x^2 n_y^2}{n_x^2 - n_y^2}\right] e_{yy} \qquad \text{(V-10)}$$

which simplifies to

$$\text{visc} = \frac{e_{xx}}{n_x^2 - n_y^2} = \frac{-e_{yy}}{n_x^2 - n_y^2}. \qquad \text{(V-11)}$$

Thus, in two dimensions, the normal stress at the interface can be determined by using the stress in the coordinate direction closest to the normal vector and geometrical considerations.

THREE-DIMENSIONAL FORMULAS

In three dimensions, the situation is a bit more complex, rather than three independent components of the Newtonian stress tensor, there are six. Therefore five additional relations would be required to eliminate all but one of the components of the stress tensor in the manner presented for two dimensions above. Unfortunately, only three additional relations are available, two orthogonal tangential stress balances and the continuity equation.

First I present equations for the normal stress, two equations for the tangential stress, and the continuity equation

$$\text{visc} = e_{xx} n_x^2 + e_{yy} n_y^2 + e_{zz} n_z^2 + 2e_{xy} n_x n_y + 2e_{yz} n_y n_z + 2e_{zx} n_z n_x, \qquad \text{(V-12)}$$

$$\begin{aligned} 0 = {} & e_{xx} n_x s_x + e_{yy} n_y s_y + e_{zz} n_z s_z \\ & + e_{xy}\left(n_x s_y + n_y s_x\right) + e_{yz}\left(n_y s_z + n_z s_y\right) + e_{zx}\left(n_z s_x + n_x s_z\right), \end{aligned} \qquad \text{(V-13)}$$

$$0 = e_{xx}n_x t_x + e_{yy}n_y t_y + e_{zz}n_z t_z + e_{xy}\left(n_x t_y + n_y t_x\right) + e_{yz}\left(n_y t_z + n_z t_y\right) + e_{zx}\left(n_z t_x + n_x t_z\right), \tag{V-14}$$

and $$0 = e_{xx} + e_{yy} + e_{zz}. \tag{V-15}$$

Since I only have three auxiliary equations, I must choose which three components of the rate of strain tensor to eliminate. If the normal is most nearly in the x-direction, then I will eliminate the components that are only in the y-z plane (the other directions are obtained by analogy). Thus, the system of equations becomes

$$\begin{bmatrix} n_x^2 & n_y^2 & n_z^2 & 2n_y n_z \\ n_x s_x & n_y s_y & n_z s_z & n_y s_z + n_z s_y \\ n_x t_x & n_y t_y & n_z t_z & n_y t_z + n_z t_y \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ e_{zz} \\ e_{yz} \end{bmatrix} = \begin{bmatrix} \text{visc} - 2e_{xy}n_x n_y - 2e_{zx}n_z n_x \\ -e_{xy}\left(n_x s_y + n_y s_x\right) - e_{zx}\left(n_z s_x + n_x s_z\right) \\ -e_{xy}\left(n_x t_y + n_y t_x\right) - e_{zx}\left(n_z t_x + n_x t_z\right) \\ 0 \end{bmatrix} \tag{V-16}$$

As is discussed elsewhere, the two unit tangent vectors can be expressed as

$$\mathbf{s} = 0\mathbf{i} + \frac{-n_z}{\sqrt{n_y^2 + n_z^2}}\mathbf{j} + \frac{n_y}{\sqrt{n_y^2 + n_z^2}}\mathbf{k} \tag{V-17a}$$

and $$\mathbf{t} = \frac{-\left(n_y^2 + n_z^2\right)}{\sqrt{n_y^2 + n_z^2}}\mathbf{i} + \frac{n_x n_y}{\sqrt{n_y^2 + n_z^2}}\mathbf{j} + \frac{n_x n_z}{\sqrt{n_y^2 + n_z^2}}\mathbf{k}. \tag{V-17b}$$

Thus, the system of equations becomes

$$\begin{bmatrix} n_x^2 & n_y^2 & n_z^2 & 2n_y n_z \\ 0 & -n_y n_z & n_z n_y & n_y^2 - n_z^2 \\ n_x\left(n_x^2 - 1\right) & n_x n_y^2 & n_x n_z^2 & 2n_y n_x n_z \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ e_{zz} \\ e_{yz} \end{bmatrix} = \begin{bmatrix} \text{visc} - 2n_x\left(e_{xy}n_y + e_{zx}n_z\right) \\ -e_{xy}\left(-n_x n_z\right) - e_{zx}\left(n_x n_y\right) \\ -\left(2n_x^2 - 1\right)\left(e_{xy}n_y + e_{zx}n_z\right) \\ 0 \end{bmatrix} \tag{V-18}$$

Dividing the third equation by n_x and subtracting it from the first equation yields

$$\left[n_x^2 - \left(n_x^2 - 1\right)\right]e_{xx} = \text{visc} - 2n_x\left(e_{xy}n_y + e_{zx}n_z\right) + \left(2n_x - 1/n_x\right)\left(e_{xy}n_y + e_{zx}n_z\right) \tag{V-19}$$

which simplifies to

$$visc = e_{xx} + \frac{n_y}{n_x} e_{xy} + \frac{n_z}{n_x} e_{zx}. \qquad \text{(V-20)}$$

By analogy, if the normal vector is nearest to the y-direction, the formula becomes

$$visc = e_{yy} + \frac{n_x}{n_y} e_{xy} + \frac{n_z}{n_y} e_{yz} \qquad \text{(V-21)}$$

and if the normal vector is nearest the z-direction, the formula becomes

$$visc = e_{zz} + \frac{n_x}{n_z} e_{xz} + \frac{n_y}{n_z} e_{yz}. \qquad \text{(V-22)}$$

This method of computing the normal stress is not implemented, and is only valid in Cartesian coordinates for Newtonian flows. The computational technique can be more easily extended to cylindrical coordinates and non-Newtonian fluid mechanics with the full stress computation currently used.

APPENDIX VI

THREE-DIMENSIONAL ANALOG TO BRAMBLE AND HUBBARD'S TECHNIQUE

Bramble and Hubbard[93] present a formulation for solving Poisson's equation having mixed boundary conditions on a region with curved boundaries. This technique is second order accurate, and presents criteria that guaranties a positive definite solution matrix. Here I present an extension of Bramble and Hubbard's technique that is applicable to three-dimensional problems.

For a given coordinate direction, s, the partial differential in terms of an alternative coordinate system consisting of the x, y, and z directions may be written as

$$\frac{\partial}{\partial s} = \frac{\partial x}{\partial s}\frac{\partial}{\partial x} + \frac{\partial y}{\partial s}\frac{\partial}{\partial y} + \frac{\partial z}{\partial s}\frac{\partial}{\partial z} \tag{VI-1}$$

and the second partial with respect to the coordinates s and t may be written as

$$\begin{aligned}\frac{\partial^2}{\partial s \partial t} = &\frac{\partial x}{\partial s}\frac{\partial}{\partial x}\left(\frac{\partial x}{\partial t}\frac{\partial}{\partial x}\right) + \frac{\partial x}{\partial s}\frac{\partial}{\partial x}\left(\frac{\partial y}{\partial t}\frac{\partial}{\partial y}\right) + \frac{\partial x}{\partial s}\frac{\partial}{\partial x}\left(\frac{\partial z}{\partial t}\frac{\partial}{\partial z}\right) \\ &+ \frac{\partial y}{\partial s}\frac{\partial}{\partial y}\left(\frac{\partial x}{\partial t}\frac{\partial}{\partial x}\right) + \frac{\partial y}{\partial s}\frac{\partial}{\partial y}\left(\frac{\partial y}{\partial t}\frac{\partial}{\partial y}\right) + \frac{\partial y}{\partial s}\frac{\partial}{\partial y}\left(\frac{\partial z}{\partial t}\frac{\partial}{\partial z}\right) \\ &+ \frac{\partial z}{\partial s}\frac{\partial}{\partial z}\left(\frac{\partial x}{\partial t}\frac{\partial}{\partial x}\right) + \frac{\partial z}{\partial s}\frac{\partial}{\partial z}\left(\frac{\partial y}{\partial t}\frac{\partial}{\partial y}\right) + \frac{\partial z}{\partial s}\frac{\partial}{\partial z}\left(\frac{\partial z}{\partial t}\frac{\partial}{\partial z}\right)\end{aligned} \tag{VI-2}$$

where s and t are again allowed to vary over any of the transformed coordinates.

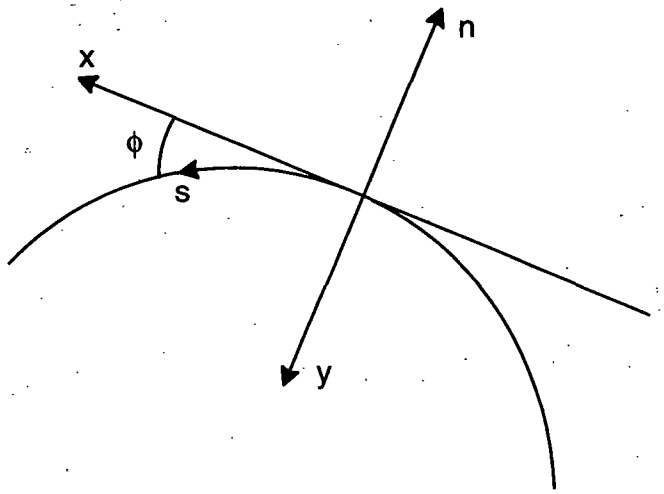

Figure IV-1. Coordinate transformation for two-dimensional interface.

In this situation, the transformation tensor relating the two coordinate systems is

$$\begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial n} & \frac{\partial y}{\partial n} & \frac{\partial z}{\partial n} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ \sin\phi & -\cos\phi\cos\theta & \sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad \text{(VI-3)}$$

which arises from geometrical considerations. Figure VI-1 shows the transformation in two-dimensions with an analogous transformation in three-dimensions yielding the matrix in Equation VI-3.

With this transformation matrix, the first partial derivatives in the transformed system become

$$v_s = \cos\phi\, v_x + \sin\phi\, v_y + 0\, v_z, \qquad \text{(VI-4)}$$

$$v_n = \sin\phi\, v_x - \cos\phi\cos\theta\, v_y + \sin\theta\, v_z, \qquad \text{(VI-5)}$$

and

$$v_t = 0\, v_x + \sin\theta\, v_y + \cos\theta\, v_z \qquad \text{(VI-6)}$$

where v is an arbitrary function and the subscripts denote differentiation. Similarly, the second partial derivatives are defined below.

First,

$$\begin{aligned} v_{ss} = {} & \cos\phi\frac{\partial}{\partial x}(\cos\phi\, v_x)+\cos\phi\frac{\partial}{\partial x}(\sin\phi\, v_y)+\cos\phi\frac{\partial}{\partial x}(0\, v_z) \\ & +\sin\phi\frac{\partial}{\partial y}(\cos\phi\, v_x)+\sin\phi\frac{\partial}{\partial y}(\sin\phi\, v_y)+\sin\phi\frac{\partial}{\partial y}(0\, v_z)\ , \\ & +0\frac{\partial}{\partial z}(\cos\phi\, v_x)+0\frac{\partial}{\partial z}(\sin\phi\, v_y)+0\frac{\partial}{\partial z}(0\, v_z) \end{aligned} \tag{VI-7}$$

which simplifies to

$$\begin{aligned} v_{ss} = {} & \cos^2\phi\, v_{xx}+\cos\phi\, v_x\frac{\partial\cos\phi}{\partial x}+\sin\phi\cos\phi\, v_{xy}+\cos\phi\, v_y\frac{\partial\sin\phi}{\partial x} \\ & +\sin\phi\cos\phi\, v_{xy}+\sin\phi\, v_x\frac{\partial\cos\phi}{\partial y}+\sin^2\phi\, v_{yy}+\sin\phi\, v_y\frac{\partial\sin\phi}{\partial y} \end{aligned} \tag{VI-8}$$

and finally,

$$\begin{aligned} v_{ss} = {} & \cos^2\phi\, v_{xx}+\sin^2\phi\, v_{yy}+2\sin\phi\cos\phi\, v_{xy} \\ & -\sin\phi\, v_x\left(\cos\phi\frac{\partial\phi}{\partial x}+\sin\phi\frac{\partial\phi}{\partial y}\right)+\cos\phi\, v_y\left(\cos\phi\frac{\partial\phi}{\partial x}+\sin\phi\frac{\partial\phi}{\partial y}\right) \end{aligned} \tag{VI-9}$$

Notice that

$$\frac{\partial\phi}{\partial s}=\cos\phi\frac{\partial\phi}{\partial x}+\sin\phi\frac{\partial\phi}{\partial y}=K \tag{VI-10}$$

Where K is the curvature in the x-y plane. Thus,

$$v_{ss}=\cos^2\phi\, v_{xx}+\sin^2\phi\, v_{yy}+2\sin\phi\cos\phi\, v_{xy}-\sin\phi\, v_xK+\cos\phi\, v_yK. \tag{VI-11}$$

By analogy,

$$v_{tt}=\cos^2\theta\, v_{zz}+\sin^2\theta\, v_{yy}+2\sin\theta\cos\theta\, v_{yz}-\sin\theta\, v_zL+\cos\theta\, v_yL \tag{VI-12}$$

where L is the curvature in the z-y plane.

For the first of the cross derivatives,

$$\begin{aligned} v_{st} &= \cos\phi\frac{\partial}{\partial x}(0\ v_x)+\cos\phi\frac{\partial}{\partial x}(\sin\theta\ v_y)+\cos\phi\frac{\partial}{\partial x}(\cos\theta\ v_z) \\ &+\sin\phi\frac{\partial}{\partial y}(0\ v_x)+\sin\phi\frac{\partial}{\partial y}(\sin\theta\ v_y)+\sin\phi\frac{\partial}{\partial y}(\cos\theta\ v_z) \\ &+0\frac{\partial}{\partial z}(0\ v_x)+0\frac{\partial}{\partial z}(\sin\theta\ v_y)+0\frac{\partial}{\partial z}(\cos\theta\ v_z) \end{aligned} \tag{VI-13}$$

which simplifies to

$$\begin{aligned} v_{st} &= \cos\phi\sin\theta\ v_{xy}+\cos\phi\cos\theta\ v_y\frac{\partial\theta}{\partial x}+\cos\phi\cos\theta\ v_{xz}-\cos\phi\sin\theta\ v_z\frac{\partial\theta}{\partial x} \\ &+\sin\phi\sin\theta\ v_{yy}+\sin\phi\cos\theta\ v_y\frac{\partial\theta}{\partial y}+\sin\phi\cos\theta\ v_{yz}+\sin\phi\sin\theta\ v_z\frac{\partial\theta}{\partial y} \end{aligned} \tag{VI-14}$$

and finally

$$\begin{aligned} v_{st} &= \cos\phi\sin\theta\ v_{xy}+\cos\phi\cos\theta\ v_{xz}+\sin\phi\sin\theta\ v_{yy}+\sin\phi\cos\theta\ v_{yz} \\ &+\cos\theta\ v_y\left(\cos\phi\frac{\partial\theta}{\partial x}+\sin\phi\frac{\partial\theta}{\partial y}\right)-\sin\theta\ v_z\left(\cos\phi\frac{\partial\theta}{\partial x}+\sin\phi\frac{\partial\theta}{\partial y}\right) \end{aligned} \tag{VI-15}$$

Since

$$\frac{\partial\theta}{\partial s}=\cos\phi\frac{\partial\theta}{\partial x}+\sin\phi\frac{\partial\theta}{\partial y}=0, \tag{VI-16}$$

the resulting equation is

$$v_{st}=\cos\phi\sin\theta\ v_{xy}+\cos\phi\cos\theta\ v_{xz}+\sin\phi\sin\theta\ v_{yy}+\sin\phi\cos\theta\ v_{yz}. \tag{VI-17}$$

The remain cross derivatives are derived next:

$$\begin{aligned} v_{sn} &= \cos\phi\frac{\partial}{\partial x}(\sin\phi\ v_x)+\cos\phi\frac{\partial}{\partial x}(-\cos\phi\cos\theta\ v_y)+\cos\phi\frac{\partial}{\partial x}(\sin\theta\ v_z) \\ &+\sin\phi\frac{\partial}{\partial y}(\sin\phi\ v_x)+\sin\phi\frac{\partial}{\partial y}(-\cos\phi\cos\theta\ v_y)+\sin\phi\frac{\partial}{\partial y}(\sin\theta\ v_z) \\ &+0\frac{\partial}{\partial z}(\sin\phi\ v_x)+0\frac{\partial}{\partial z}(-\cos\phi\cos\theta\ v_y)+0\frac{\partial}{\partial z}(\sin\theta\ v_z) \end{aligned} \tag{VI-18}$$

which simplifies to

$$
\begin{aligned}
v_{sn} &= \sin\phi\cos\phi\, v_{xx} - \sin\phi\cos\phi\cos\theta\, v_{yy} + \left(\sin^2\phi - \cos^2\phi\cos\theta\right)v_{xy} \\
&\quad + \cos\phi\sin\theta\, v_{xz} + \sin\phi\sin\theta\, v_{yz} \\
&\quad + \cos\phi\, v_x\left(\cos\phi\frac{\partial\phi}{\partial x} + \sin\phi\frac{\partial\phi}{\partial y}\right) + \sin\phi\cos\theta\, v_y\left(\cos\phi\frac{\partial\phi}{\partial x} + \sin\phi\frac{\partial\phi}{\partial y}\right) \\
&\quad + \cos\theta\, v_z\left(\cos\phi\frac{\partial\theta}{\partial x} + \sin\phi\frac{\partial\theta}{\partial y}\right) + \cos\phi\sin\theta\, v_y\left(\cos\phi\frac{\partial\theta}{\partial x} + \sin\phi\frac{\partial\theta}{\partial y}\right)
\end{aligned}
\tag{VI-19}
$$

and finally

$$
\begin{aligned}
v_{sn} &= \sin\phi\cos\phi\, v_{xx} - \sin\phi\cos\phi\cos\theta\, v_{yy} + \left(\sin^2\phi - \cos^2\phi\cos\theta\right)v_{xy} \\
&\quad + \cos\phi\sin\theta\, v_{xz} + \sin\phi\sin\theta\, v_{yz} + \cos\phi\, v_x K + \sin\phi\cos\theta\, v_y K
\end{aligned}
\tag{VI-20}
$$

Again, by analogy:

$$
\begin{aligned}
v_{tn} &= \sin\theta\cos\theta\, v_{zz} - \cos\phi\sin\theta\cos\theta\, v_{yy} + \left(\sin^2\theta - \cos\phi\cos^2\theta\right)v_{yz} \\
&\quad + \sin\phi\cos\theta\, v_{xz} + \sin\phi\sin\theta\, v_{xy} + \cos\theta\, v_z L + \cos\phi\sin\theta\, v_y L
\end{aligned}
\tag{VI-21}
$$

In summary and incorporating the knowledge that $\phi = \theta = 0$ at the point of interest

$$v_s = v_x, \tag{VI-22a}$$

$$v_t = v_z, \tag{VI-22b}$$

$$v_n = -v_y, \tag{VI-22c}$$

$$v_{ss} = v_{xx} + v_y K, \tag{VI-22d}$$

$$v_{tt} = v_{zz} + v_y L, \tag{VI-22e}$$

$$v_{sn} = -v_{xy} + v_x K, \tag{VI-22f}$$

$$v_{tn} = -v_{yz} + v_z L, \tag{VI-22g}$$

and $$v_{st} = v_{xz}. \tag{VI-22h}$$

One additional relation,

$$\nabla^2 v = \Delta v = v_{xx} + v_{yy} + v_{zz}, \tag{VI-23}$$

is needed.

The relations in (VI-22) and (VI-23) may be rearranged to yield

$$v_x = v_s, \tag{VI-24a}$$

$$v_z = v_t, \tag{VI-24b}$$

$$v_y = -v_n, \tag{VI-24c}$$

$$v_{xx} = v_{ss} + v_n K, \tag{VI-24d}$$

$$v_{zz} = v_{tt} + v_n L, \tag{VI-24e}$$

$$v_{xy} = -v_{sn} + v_s K, \tag{VI-24f}$$

$$v_{yz} = -v_{tn} + v_t L, \tag{VI-24g}$$

$$v_{xz} = v_{st}, \tag{VI-24h}$$

and $$v_{yy} = \Delta v - v_{xx} - v_{zz}. \tag{VI-24i}$$

I will also need a three-dimensional Taylor series expansion of v about point 0

$$\begin{aligned} v(P) = v(0) &+ x\, v_x(0) + y\, v_y(0) + z\, v_z(0) \\ &+ \tfrac{1}{2}\left(x^2\, v_{xx}(0) + y^2\, v_{yy}(0) + z^2\, v_{zz}(0)\right) \\ &+ \left(xy\, v_{xy}(0) + xz\, v_{xz}(0) + yz\, v_{yz}(0)\right) \end{aligned} \tag{VI-25}$$

GENERATION OF A FIRST ORDER ACCURATE FORMULA

For the first order accurate finite difference formula, I begin with the three-dimensional Taylor series expansion neglecting the second derivatives, and substitute the appropriate relations from (VI-24),

$$v(P) = v(0) + x\, v_s(0) - y\, v_n(0) + z\, v_t(0). \tag{VI-26}$$

Multiplying by b_i and summing over i = 1 to 3 yields

$$\sum_{i=1}^{3} b_i v(P_i) = \sum_{i=1}^{3} b_i v(0) + \sum_{i=1}^{3} b_i x_i v_s(0) + \sum_{i=1}^{3} b_i z_i v_t(0) - \sum_{i=1}^{3} b_i y_i v_n(0) \tag{VI-27}$$

Since I am only interested in the normal derivative, I can define three equation to eliminate the terms containing the tangential derivatives and set the normal derivative equal to unity.

$$\sum_{i=1}^{3} b_i y_i = 1, \tag{VI-28a}$$

$$\sum_{i=1}^{3} b_i x_i = 0, \tag{VI-28b}$$

and

$$\sum_{i=1}^{3} b_i z_i = 0. \tag{VI-28c}$$

Thus equation (VI-27) reduces to

$$\begin{aligned} \delta_n v(0) &= \sum_{i=1}^{3} b_i \left[v(0) - v(P_i) \right] \\ &= v_n(0) \\ &= g(0). \end{aligned} \tag{VI-29}$$

which after rearranging to solve for the value at the interface yields

$$v(0) = \frac{g(0) + \sum_{i=1}^{3} b_i v(P_i)}{\sum_{i=1}^{3} b_i}. \tag{VI-30}$$

GENERATION OF A SECOND ORDER ACCURATE FORMULA

The second order accurate formula is derived in the same manner as the first order accurate formula described above. I begin with the three-dimensional Taylor series in (VI-25), which after substituting the relations in (VI-24) yields

$$\begin{aligned} v(P) = v(0) &+ x\, v_s(0) - y\, v_n(0) + z\, v_t(0) + \tfrac{1}{2} y^2\, \Delta v \\ &+ \tfrac{1}{2}\left(x^2 - y^2\right)\left(v_{ss}(0) + v_n(0)K(0)\right) \\ &+ \tfrac{1}{2}\left(z^2 - y^2\right)\left(v_{tt}(0) + v_n(0)L(0)\right) \\ &+ xy\left(-v_{sn}(0) + v_s(0)K(0)\right) + xz\, v_{st}(0) + yz\left(-v_{tn}(0) + v_t(0)L(0)\right) \end{aligned} \tag{VI-31}$$

which may be rearranged by combining terms to

$$\begin{aligned} v(P) = v(0) &+ x\left(1 + y\, K(0)\right)v_s(0) + z\left(1 + y\, L(0)\right)v_t(0) \\ &- \left(y - \tfrac{1}{2}\left(x^2 - y^2\right)K(0) - \tfrac{1}{2}\left(z^2 - y^2\right)L(0)\right) v_n(0) \\ &+ \tfrac{1}{2}\left(x^2 - y^2\right) v_{ss}(0) + \tfrac{1}{2}\left(z^2 - y^2\right)v_{tt}(0) + \tfrac{1}{2} y^2\, \Delta v \\ &- xy\, v_{sn}(0) - yz\, v_{tn}(0) + xz\, v_{st}(0). \end{aligned} \tag{VI-32}$$

Multiplying by a_i and summing over i = 1 to 6 yields

$$\begin{aligned} \sum_{i=1}^{6} a_i v(P_i) = &\sum_{i=1}^{6} a_i v(0) + \sum_{i=1}^{6} a_i x_i \left[1 + y_i K(0)\right]v_s(0) + \sum_{i=1}^{6} a_i z_i \left[1 + y_i L(0)\right]v_t(0) \\ &- \sum_{i=1}^{6} a_i \left[y_i - \tfrac{1}{2}\left(x_i^2 - y_i^2\right)K(0) - \tfrac{1}{2}\left(z_i^2 - y_i^2\right)L(0)\right]v_n(0) \\ &+ \tfrac{1}{2}\sum_{i=1}^{6} a_i \left(x_i^2 - y_i^2\right)v_{ss}(0) + \tfrac{1}{2}\sum_{i=1}^{6} a_i \left(z_i^2 - y_i^2\right)v_{tt}(0) + \tfrac{1}{2}\sum_{i=1}^{6} a_i y_i^2 \Delta v \\ &- \sum_{i=1}^{6} a_i x_i y_i v_{sn}(0) - \sum_{i=1}^{6} a_i y_i z_i v_{tn}(0) + \sum_{i=1}^{6} a_i x_i z_i v_{st}(0). \end{aligned} \tag{VI-33}$$

In order to derive a second order accurate formula, I need to eliminate the terms containing v_s, v_t, v_{ss}, v_{tt}, and v_{st} and set the term containing v_n=1. Thus there are six simultaneous equations to solve:

$$\sum_{i=1}^{6} a_i\left[y_i - \tfrac{1}{2}\left(x_i^2 - y_i^2\right)K(0) - \tfrac{1}{2}\left(z_i^2 - y_i^2\right)L(0)\right] = \sum_{i=1}^{6} a_i y_i = 1, \qquad \text{(VI-34a)}$$

$$\sum_{i=1}^{6} a_i x_i\left(1 + y_i K(0)\right) = 0, \qquad \text{(VI-34b)}$$

$$\sum_{i=1}^{6} a_i z_i\left(1 + y_i L(0)\right) = 0, \qquad \text{(VI-34c)}$$

$$\sum_{i=1}^{6} a_i\left(x_i^2 - y_i^2\right) = 0, \qquad \text{(VI-34d)}$$

$$\sum_{i=1}^{6} a_i\left(z_i^2 - y_i^2\right) = 0, \qquad \text{(VI-34e)}$$

and $$\sum_{i=1}^{6} a_i x_i z_i = 0. \qquad \text{(VI-34f)}$$

Which, assuming x_i, y_i, and z_i to be small, is approximated by the system:

$$\sum_{i=1}^{6} \bar{a}_i y_i = 1, \qquad \text{(VI-35a)}$$

$$\sum_{i=1}^{6} \bar{a}_i x_i = 0, \qquad \text{(VI-35b)}$$

$$\sum_{i=1}^{6} \bar{a}_i z_i = 0, \qquad \text{(VI-35c)}$$

$$\sum_{i=1}^{6} \bar{a}_i\left(x_i^2 - y_i^2\right) = 0, \qquad \text{(VI-35d)}$$

$$\sum_{i=1}^{6} \bar{a}_i\left(z_i^2 - y_i^2\right) = 0, \qquad \text{(VI-35e)}$$

and $$\sum_{i=1}^{6} \bar{a}_i x_i z_i = 0. \qquad \text{(VI-35f)}$$

Thus, Equation (VI 33) reduces to:

$$\begin{aligned}\delta_n v(0) &= \sum_{i=1}^{6} a_i\left[v(0) - v(P_i)\right] \\ &= v_n(0) - \tfrac{1}{2}\sum_{i=1}^{6} a_i y_i^2 \Delta v + \sum_{i=1}^{6} a_i x_i y_i v_{sn}(0) + \sum_{i=1}^{6} a_i y_i z_i v_{tn}(0) \qquad \text{(VI-36)} \\ &= g(0) + \sum_{i=1}^{6} a_i\left[x_i y_i g_s(0) + y_i z_i g_t(0)\right]\end{aligned}$$

where $g(0)$ is the normal derivative at the point of interest, $g_s(0)$ is the curvature in the x-y plane, and $g_t(0)$ is the curvature in the y-z plane.

Rearranging (VI 36) yields

$$v(0)=\frac{\sum_{i=1}^{6} a_i v(P_i)}{\sum_{i=1}^{6} a_i}+\frac{1}{\sum_{i=1}^{6} a_i} g(0)+\frac{\sum_{i=1}^{6} a_i x_i y_i}{\sum_{i=1}^{6} a_i} g_s(0)+\frac{\sum_{i=1}^{6} a_i y_i z_i}{\sum_{i=1}^{6} a_i} g_t(0) \qquad \text{(VI-37)}$$

for the value of the arbitrary function at the boundary in terms of neighboring points and the normal derivative.

Thus, In order to apply the boundary condition in three-dimensions, six points within the domain must be chosen such that linear system in (VI-35) can be solved to yield all positive values for the coefficients, a_i. Equation (VI-37) may then be used to determine the boundary value.

APPENDIX VII

NUMERICAL SOLUTION OF LI AND TANKIN'S DISPERSION RELATIONS

In their analysis of the stability of a thin viscous sheet flowing through an inviscid vapor phase, Li and Tankin[58] present dispersion relations linking the growth rate of waves in the sheet of liquid with the wavenumber of the disturbance. These relations are

$$0 = \left(\tilde{\omega}_1 + 4m^2Z\right)\tilde{\omega}_1 \tanh(m) + 4m^3Z^2\left[m\tanh(m) - \left(m^2 + \tilde{\omega}_1/Z\right)^{\frac{1}{2}} \tanh\left(m^2 + \tilde{\omega}_1/Z\right)^{\frac{1}{2}}\right] + \tilde{\rho}\tilde{\omega}^2 + m^3 \quad \text{(VII-1)}$$

and

$$0 = \left(\tilde{\omega}_1 + 4m^2Z\right)\tilde{\omega}_1 \coth(m) + 4m^3Z^2\left[m\coth(m) - \left(m^2 + \tilde{\omega}_1/Z\right)^{\frac{1}{2}} \coth\left(m^2 + \tilde{\omega}_1/Z\right)^{\frac{1}{2}}\right] + \tilde{\rho}\tilde{\omega}^2 + m^3 \quad \text{(VII-2)}$$

for antisymmetric and axisymmetric disturbances, respectively. The parameters in these equations are defined in the text above, but I will repeat the important ones here:

$\tilde{\omega}$, the dimensionless complex growth rate

$\tilde{\omega}_1 = \tilde{\omega} + i\, We_\ell^{\frac{1}{2}} m$,

$m = ka$, the dimensionless wave number

$We_\ell = We_v/\tilde{\rho}$, the liquid phase Weber number

$We_v = \rho_v U_0^2 a/\sigma$, the gas phase Weber number

$\tilde{\rho} = \rho_v/\rho_\ell$, the density ratio

$Z = \mu_\ell/\sqrt{\rho_\ell a\sigma}$, the Ohnesorge number.

Below is a source code listing of the FORTRAN program used to solve this extremely non-linear problem involving complex numbers. The program begins by initializing the variables including the gas phase Weber number, the Ohnesorge number and the density ratio. Next the program loops over the desired range of wavenumbers ad solves Equation (VII-1) and (VII-2) for the growth rate at that wavenumber. Four cases are considered for the axisymmetric and antisymmetric cases: the complete equation;

neglecting surface tension, the term m^3; neglecting the vapor phase, the term $\tilde{\rho}\tilde{\omega}^2$; and neglecting both the vapor phase and surface tension, the terms $\tilde{\rho}\tilde{\omega}^2$ and m^3.

The solution procedure is to used direct substitution and treat the complex growth rate as a single equation and a single unknown. Thus we begin with an initial guess of (0.0,0.0) as the complex growth rate, substitute this into the equation being solved, yielding a residual that should be near zero. If the residual is not small enough it is subtracted (with an under-relaxation factor) from the previous value of the complex growth rate and the process is repeated.

Table VII-1. Source code listing for computation of the complex growth rate.

```
      program omega
      implicit none
      integer j, l
      double complex omg(8), eps, omg1, s, ztanh
      double precision m, WEg, Z, rhor
c
      ztanh(s)=(zexp(s)-zexp(-s))/(zexp(s)+zexp(-s))
c
      WEg=4.0d0
      Z=0.1d0
      rhor=0.1d0
c
      write(*,998) Z, rhor, WEg
      do l=1,100
        m=5.0d0/100.0d0*float(l)
c
c
c     antisymmetric
c
        eps=(0.0d0,0.0d0)
        omg(1)=eps
        do j=1,100000
          omg(1)=omg(1)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(1)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1*tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m*tanh(m)-s*ztanh(s))
     2       + rhor*omg(1)*omg(1)+m*m*m
          if (zabs(eps).lt.1.0d-10) go to 111
        enddo
  111   continue
c
c     axisymmetric
c
        eps=(0.0d0,0.0d0)
        omg(2)=eps
        do j=1,100000
          omg(2)=omg(2)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(2)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1/tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m/tanh(m)-s/ztanh(s))
     2       + rhor*omg(2)*omg(2)+m*m*m
          if (zabs(eps).lt.1.0d-10) go to 222
        enddo
  222   continue
c
c     antisymmetric w/o surface tension
c
        eps=(0.0d0,0.0d0)
        omg(3)=eps
```

```
        do j=1,100000
          omg(3)=omg(3)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(3)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1*tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m*tanh(m)-s*ztanh(s))
     2       + rhor*omg(3)*omg(3)
          if (zabs(eps).lt.1.0d-10) go to 333
        enddo
  333   continue
c
c     axisymmetric w/o surface tension
c
        eps=(0.0d0,0.0d0)
        omg(4)=eps
        do j=1,100000
          omg(4)=omg(4)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(4)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1/tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m/tanh(m)-s/ztanh(s))
     2       + rhor*omg(4)*omg(4)
          if (zabs(eps).lt.1.0d-10) go to 444
        enddo
  444   continue
c
c     antisymmetric w/o vapor phase
c
        eps=(0.0d0,0.0d0)
        omg(5)=eps
        do j=1,100000
          omg(5)=omg(5)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(5)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1*tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m*tanh(m)-s*ztanh(s))
     2       + m*m*m
          if (zabs(eps).lt.1.0d-10) go to 555
        enddo
  555   continue
c
c     axisymmetric w/o vapor phase
c
        eps=(0.0d0,0.0d0)
        omg(6)=eps
        do j=1,100000
          omg(6)=omg(6)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(6)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1/tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m/tanh(m)-s/ztanh(s))
     2       + m*m*m
          if (zabs(eps).lt.1.0d-10) go to 666
        enddo
```

```
  666   continue
c
c     antisymmetric w/o vapor phase w/o surface tension
c
        eps=(0.0d0,0.0d0)
        omg(7)=eps
        do j=1,100000
          omg(7)=omg(7)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(7)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1*tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m*tanh(m)-s*ztanh(s))
          if (zabs(eps).lt.1.0d-10) go to 777
        enddo
  777   continue
c
c     axisymmetric w/o vapor phase w/o surface tension
c
        eps=(0.0d0,0.0d0)
        omg(8)=eps
        do j=1,100000
          omg(8)=omg(8)-0.001d0/m*(1.0d0,1.0d0)*eps
          omg1=omg(8)+(0.0d0,m*sqrt(WEg/rhor))
          s=zsqrt(m*m+omg1/Z)
          eps=(omg1+4.0d0*m*m*Z)*omg1/tanh(m)
     1       + 4.0d0*m*m*m*Z*Z*(m/tanh(m)-s/ztanh(s))
          if (zabs(eps).lt.1.0d-10) go to 888
        enddo
  888   continue
        write(*,999) m, (dreal(omg(j)),j=1,8)
      enddo
c
  998 format(1p,5e14.6)
  999 format(f10.8,1p,8e14.6)
      return
      end
```

APPENDIX VIII

BUGS IN THE NASA-VOF3D PROGRAM

During the course of the development of the IPST-VOF3D computational technique, I have discovered several programming errors or "bugs" in the NASA-VOF3D implementation. These bugs will be presented by the incorrect line in the NASA-VOF3D program, the line number in the source code listing of Torrey et al.,[81] and the corrected source line.

NO-SLIP BOUNDARY CONDITIONS IN **BC** ASSUME STATIONARY WALLS

This error occurs in many places in the **BC** subroutine, but only one is presented as an example.

```
                                                          line
                                                         number
v(ijk-1)=-v(ijk)*delxrl                                    49
w(ijk-1)=-w(ijk)*delxrl                                    50

v(ijk-1)=vlw+(vlw-v(ijk))*delxrl
w(ijk-1)=wlw+(wlw-w(ijk))*delxrl
```

The additional variables VLW and WLW indicate the velocities of the domain boundaries. VLW refers to the v-velocity component on the left boundary and WLW refers to the w-velocity component on the left boundary. Similarly, VTW would refer to the v-velocity component of the top boundary.

ERROR IN **BCFS** RELATING TO CARTESIAN COORDINATES

These statements were valid only for problems in cylindrical coordinates and led to errors in certain Cartesian problems (note that the NASA-VOF3D program was only intended for cylindrical problems).

```
      if (ar(ijk).gt.em6) u(ijk)=u(imjk)*ar(imjk)*x(i-1)/(ar(ijk)*x(i))     42
      if (ar(imjk).gt.em6) u(imjk)=u(ijk)*ar(ijk)*x(i)/(ar(imjk)*x(i-1))    48

       if (ar(ijk).gt.em6) u(ijk)=u(imjk)
     1  *(1.0-cyl+cyl*ar(imjk)*x(i-1)/(ar(ijk)*x(i)))
       if (ar(imjk).gt.em6) u(imjk)=u(ijk)
     1  *(1.0-cyl+cyl*ar(ijk)*x(i)/(ar(imjk)*x(i-1)))
```

ERROR IN COMPUTATION OF THE SOR PARAMETER, β, IN **BETACAL**

The equations for computing the SOR parameter β is incorrect. The term rri(i) occurs an extra time in line 32. It appears that it was factored out of the expression containing abbk and abf, but not removed from the expression.

```
      xx=2.0*delt*(rdx(i)*(abr/(delx(i)+delx(i+1))+abl/(delx(i)+delx(i-1    30
     1 )))+rdy(j)*rri(i)*(abbk*rri(i)/(dely(j)+dely(j+1))+abf*rri(i)/       31
     2 (dely(j)+dely(j+1)))+rdz(k)*(abt/(delz(k)+delz(k+1))+abb/(delz(k)    32
     3 +delz(k-1)))+cyl*0.5*(abr/(delx(i)+delx(i+1))-abl/(delx(i-1)+delx    33
     4 (i)*rri(i)/x(im1))                                                  34
       beta(ijk)=omg/xx*ac(ijk)                                            35

       beta(ijk)=omg*ac(ijk)/(delt*
     1  (rdx(i)*(abr*rdxp(i)+abl*rdxp(i-1))
     2  +rdy(j)*rri(i)*(abbk*rdyp(j)+abf*rdyp(j-1))
     3  +rdz(k)*(abt*rdzp(k)+abb*rdzp(k-1))
     4  +cyl*0.5d0*(abr*rdxp(i)-abl*rdxp(i-1))*rxi(i)))
```

POSSIBLE DIVISION BY ZERO IN **MESHX**, **MESHY**, AND **MESHZ**

In the subroutines **MESHX**, **MESHY**, and **MESHZ** there is code that can lead to division by zero in single precision implementations. The section of code for **MESHX** is presented with analogous changes required in **MESHY** and **MESHZ**.

```
      dxm1=(xc(1)-xl(1))/nxl(1)                                          10
      dxmn1=dxmn(1)                                                      11
      nt=nxl(1)                                                          12
      tn=nt                                                              13
      tn=amaxz(tn,1.0+1.0d-14)                                           14
      dxmn(1)=amin1(dxmn1,dxm1)                                          15
      cmc=(xc(1)-xl(1)-tn*dxmn(1))*tn/(tn-1.0)                           16
      if (nt.eq.1) cmc=0.0                                               17
```

On a single precision implementation, the term 1.0+1.0e-14 in line 14 could be rounded to 1.0 yielding a division by zero in line 16. This section of the program was restructured to eliminate the division when nt=1 since cmc is set to zero anyway.

```
tn=float(nxl(l))
dxml=(xc(l)-xl(l))/tn
dxmn1=dxmn(l)
dxmn(l)=dmin1(dxmn1,dxml)
if (nxl(l).eq.1) then
  cmc=0.0d0
else
  cmc=(xc(l)-xl(l)-tn*dxmn(l))*tn/(tn-1.0d0)
endif
```

INCORRECT SUBROUTINE CALL IN **SETUP**

The NASA-VOF3D program called specific subroutines to compute the indices corresponding to a computational cell and its neighbors. For efficiency, these subroutines are no longer called explicitly, but the indices are computed as needed. In the subroutine **SETUP**, the index initialization subroutine IJKONLY was called where IJKAJCT was required.

```
call ijkonly                                                    133
```

should have been

```
call ijkajct                                                    133
```

As mentioned, this bug was fixed when the calls to the index computation subroutines were replaced.

BUG IN **SURF10N**

The improper index was changed.

```
if (wbk.eq.4) m=2                                               432
```

```
if (wbk.eq.4) mm=2
```

APPENDIX IX

IPST-VOF3D SOURCE CODE LISTING

The listing of the FORTRAN source code for the IPST-VOF3D program is available for viewing in the Haselton Library at the Institute of Paper Science and Technology in Atlanta, Georgia.