Title
A Unified Numerical Model for Pool Boiling Curve with Parallel Computing

Author
Garg, Deepak

Publication Date
2017

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

A Unified Numerical Model for Pool Boiling Curve with Parallel Computing

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Mechanical Engineering

by

Deepak Garg

2017

ABSTRACT OF THE DISSERTATION

A Unified Numerical Model for Pool Boiling Curve with Parallel Computing

by

Deepak Garg

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2017

Professor Vijay K. Dhir, Chair

Boiling heat transfer research spans several decades with extensive data accumulation from several experiments. Several mechanistic models and empirical correlations have also been put forth but their applicability is limited to the narrow range of parameters over which they have been developed. For the past few decades several numerical methods have also been developed and gained considerable momentum to study boiling process. The most popular of these numerical methods are volume of fluid method, level set method and front tracking methods, however some other computational methods like Lattice Boltzmann, Moving particle semi-implicit gridless and cellular automata SIMPLER methods have also been developed.

In the present study level set method is used to simulate the entire boiling curve in a temperature controlled mode spanning all the three regimes *viz.* nucleate, transition and film boiling with a unified numerical model supplemented with correlations specifying nucleation site density and bubble waiting time. In order to improve the performance of the code parallel computing has also been implemented. Both two-dimensional and three-dimensional simulations have been done for saturated water with different contact angles for a horizontal surface with uniform wall superheat applied to it. Temporal and spatial averaged wall heat flux and wall void fraction computed for a fixed wall superheat case are plotted and analyzed. For a specified contact angle and by incrementing the wall superheat as two independent input parameters, the entire boiling curve along with vapor removal patterns capturing its vital points like the maximum heat flux and minimum heat flux are shown.

The two-dimensional assumption yields opposite trend for the nucleate boiling regime heat flux variation with contact angle but this anomaly was not observed in the three-dimensional simulations which infers that the two-dimensional assumption is an incorrect representation to study the essential physics of the boiling process. The trend of critical heat flux with contact angle was found to be decreasing with increase in contact angle for both two-dimensional and three-dimensional case, with the trend being steeper for the former.

Wall void fraction was found to increase with increase in wall superheat as different regimes of boiling were traversed, and also with increase in contact angle at a given wall superheat. Mushroom type vapor bubbles are seen in the nucleate boiling regime with liquid macrolayer

trapped underneath it while long column of sustained vapor is seen at the critical heat flux condition continuously being fed by nucleating cavities at the surface. Upon increasing the wall superheat beyond critical heat flux the negative slope of the boiling curve is captured characterized by the transition boiling regime with intermittent liquid solid contacts seen. Finally, the transition to film boiling is seen with entire surface covered with superheated vapor and wall void fraction reaching unity. Energy partitioning from wall into liquid, interface and microlayer has also been examined for the 3D cases. For the 3D coarse grid case it was found that, as the wall void fraction increases the percent energy going into liquid decreases from about 85% in lower nucleate boiling to 6% in film boiling while the microlayer contribution peaks around CHF to a value of about 45%. The energy partition from the fine grid cases were however inconclusive as they couldn`t be run long enough to obtain any meaningful results.

The dissertation of Deepak Garg is approved.

Adrienne Lavine

Laurent Pilon

Tajendra V. Singh

Warren Mori

Vijay K. Dhir, Committee Chair

University of California, Los Angeles

2017

Dedicated to my wife, Neha Doshi Garg

Table of Contents

List of Figures

List of Tables

Nomenclature

A	matrix
Ar	Archimedes Number
C_2	constant
F_L	fractional area occupied by liquid
g	gravitational acceleration, m/s^2
h	grid height
$\bar{h}$	average heat transfer coefficient, W/K
h_{fg} ,h_{lv}	latent heat of vaporization, W/kg
H	domain height, m, Heaviside function
I	interface
k	thermal conductivity, W/m-k
l	length,m
L	sub-matrix
p	pressure, N/m^2
q	heat flux,W/m^2
t	time, s
T	temperature, K
$\dot{m}$	mass flux, kg/m^2
n	normal
M	matrix
N	number of cavities

Na	nucleation site density, sites/cm^2
Nu	Nusselt number
R_0	radial distance from the cavity center to the microlayer starting point, m
R_l	radial distance from the cavity center to the microlayer end point, m
u,v,w	velocity, m/s
V	volume, m^3
x,y,z	dimensionless length

Greek symbols

Δx	grid spacing in x direction
Δy	grid spacing in y direction
λ_D	Taylor`s dangerous wavelength,m
τ	pseudo time
ρ	density, kg/m^3
Φ	level set function
δ_T	thermal layer thickness, m
δ	microlayer thickness, m
ξ	distance, m
δ_0	initial microlayer thickness, m
φ	contact angle, º
β	constant, ratio of sensible heat to latent heat
β_T	coefficient of thermal expansion,1/K

α	thermal diffusivity, m^2/s, wall void fraction
σ	surface tension, N-m
μ	dynamic viscosity, Pa-s
Γ	interface
Ω	phase domain
ϵ	element of
κ	curvature
Λ	coefficient matrix
ι	evaporation coefficient
ΔV_{micro}	Vapor side control volume near micro region

Subscripts

0	initial
atm	atmospheric
b	black
bub	bubble
col	columns
d	dry
e,w,n,s	east, west, north and south
eff	effective
ev	evaporation
f	phase

l,liq	liquid
n	old time step
n+1	new time step
nc	natural convection
o	characteristic
P	principal cell
p	processor
r	red
sat	saturated
T	thermal,transpose
int	interface
micro	microlayer
min	minimum
max	maximum
v,vap	vapor
wait	waiting
wall	wall
y	y direction
X	X direction
zub	zuber
*,**	dimensionless, intermediate value

Superscripts

G	ghost cell
L	user specified

Acknowledgements

I would like to express sincere gratitude to my advisor Professor Vijay K Dhir for giving me the opportunity to work in this field along with his constant motivation, enthusiasm and guidance throughout my research at UCLA. Financial support from NASA is greatly acknowledged. I would also like to thank Dr. Tajendra V Singh for his patience and incessant mentoring for the parallelization project. The support and help given to me by Charlie Kawczynski, Dr. Mustafa Kilic, Dr. Lokanath Mohanta, Dr. Tung Vu and Haojie Huang is highly appreciated. Finally, I would like to thank my family members for their encouragement and support throughout my academics.

Vita

2006	Bachelors in Marine Engineering, Jadavpur University Kolkata, WB, India
2012	Masters in Mechanical Engineering, Purdue University West Lafayette, IN, USA
2015	Masters in Aerospace Engineering, UCLA Los Angeles, CA, USA

Publications

Deepak Garg, "Controlling Microbial Degradation of Diesel Engine Oil using Magnetic Field", *International Conference on Recent Advances of in Marine Antifouling Technology* ,NIOT Chennai , pp. 400-408, November 2006

Deepak Garg, Dr S Chatterjee, "Novel Mechanism for Athermalization of Thermographic Camera using Thermoelectrics", *19th National and 8th International ISHMT-ASME HMT Conference,* JNTUHyderabad, INDIA, January 2008.

Dr S Chatterjee, Dr P K Purkait, **Deepak Garg**, "Athermalization of Infra-Red Camera of Projectile Weapons" *,Journal of Applied Thermal Engineering*, Vol. 29, Issue 10, pp. 2106-2112, July 2009.

Deepak Garg, V K Dhir, "Two Dimensional Numerical Simulation of Boiling Curve", 9th International Conference on Boiling and Condensation, USA, April 2015.

.

1. Introduction

Boiling is a process of turning liquid at its saturation temperature into vapor by applying heat, similar to flashing where liquid is converted to vapor by reducing the pressure. Boiling heat transfer is a widespread phenomenon and has myriad applications ranging from cooking, power generation to industrial process control. Heat transfer by virtue of boiling is the preferred mode in both conventional and nuclear power plants as single phase methods including both natural convection and forced convection renders the technique incapable of meeting cooling requirements. Apart from that, boiling is also used to cool electronic components in power electronics industry and in refrigeration industry as well. Boiling heat transfer has significantly higher heat transfer coefficient than single phase heat transfer. Enhanced understanding of boiling process, in terms of all the dependent parameters and limitations, is essential for optimized technological applications.

An efficient control of the boiling process can be achieved by highly resolved experiments or by high fidelity numerical simulations achievable using high performance computing. Some of the physics behind boiling still remains to be elusive owing to the small length and time scales associated with the boiling process. For the past several decades numerous researchers have studied this process theoretically, experimentally and numerically. Analytical solutions exist only for simple problems like steady-state motion of bubbles and droplets in stokes flow and oscillations of bubbles and droplets. From the mathematical point of view of modeling of boiling, which is one form of multiphase flow, apart from non-linear nature of the governing equation and

interacting sub-processes, accurate tracking of the phase boundary is also an elusive task.

Nukiyama (1934) published one of the most phenomenal work in boiling heat transfer by plotting the heat transferred to the boiling water over a large range of wall superheats. Nukiyama`s boiling curve has never been disputed but the shape has been the subject of discussion especially with respect to transient or steady-state conditions. A typical boiling curve is shown in Figure 1 showing different boiling regimes obtained in a temperature controlled mode.

Initially, at low wall superheats heat transfer occurs only by natural convection. The wall superheat is higher than the saturation temperature of the liquid but not high enough to activate any nucleation site. Upon increasing the wall superheat the first of the bubbles nucleate at the surface and this point is called the onset of nucleate boiling (ONB). After the onset of boiling isolated bubbles grow and eventually detach the boiling surface without interacting with each other with increase in wall heat flux. Under high heat flux conditions vertical coalescence of the bubbles occurs forming vapor columns and the lateral coalescence of these vapor columns results in the formation of a vapor mass below which a liquid macrolayer is entrapped. The thickness of the macrolayer decreases with time while the vapor mass grows. As the size of the nucleating cavity varies inversely with wall superheat, as the wall superheat is increased more nucleation sites are activated augmenting bubble density. With the bubbles interacting with each other leading to further enhancement of heat flux occurs ultimately reaching upto a point where any further increase in wall superheat results in the decrease of wall heat flux, known as the point of critical heat flux (CHF). This region marked by the negative slope is known as the transition boiling regime and continues upto the point where a stable vapor film between the liquid and the wall is formed known as the film boiling, which corresponds to point c of Figure 1.

Typically an efficient cooling mechanism should operate in the nucleate boiling regime where high heat flux can be removed at moderate wall superheats. Minimization of the wall superheat is one of the optimal points of operation in power generation cycle or cooling device. In some instances like spray cooling and quenching process film boiling is used to achieve desired heat transfer rates. One of the most important aspect of boiling heat transfer is the 3-phase contact line. The local heat flux at the 3-phase contact line reaches a maximum value. It has been reported that during the advancing motion of the contact line there is an increase in the local heat transfer. However, the temporal and spatial resolution of the different measurement techniques like Thermochromic Liquid Crystals, High Speed Infra Red Thermography and Micro heater arrays, etc are still short of unraveling the microscale heat transfer mechanism in the vicinity of the 3-phase contact line.

Numerical simulations to study boiling offers promising avenue and several numerical methods have been developed for simulation of incompressible two phase flows with phase change. The past couple of decades have witnessed widespread implementation of different phase change models into a variety of novel and traditional computational schemes. Numerical modeling have improved considerably in terms of spatial and temporal resolution and capturing the physics across the interface thus increasing the overall accuracy. One classical approach is to employ moving mesh methods so that the interface coincides with the grid points. Although matching conditions at the interface in these methods are easy to implement yet they become very complicated for merging and breaking of interfaces. A numerical method on fixed grid can overcome merging and breaking of interfaces but matching conditions at the interface gets difficult. Lagrangian based methods are not easy to implement for problems involving changes in interface

topology. Surface tension on the moving interface must be considered to account for capillary effects and other curvature related phenomenon. In the presence of interfacial tension, the algorithm should be able to obtain and follow an interface including merging and breaking. To treat surface tension effects the continuum surface force (CSF) model is widely used. The number of methods can be divided into different categories like front tracking (Juric and Tryggvason (1998), volume of fluid (Hirt and Nicolas, (1981)), level set method (Osher and Sethian (1988)), and hybrid methods (McKee *et al*, (2008)). Gibou *et al.*(2007) used level set method to capture the interface evolution using ghost fluid method. Discontinuity nature of material properties like density and viscosity are tracked and implemented using level set function.

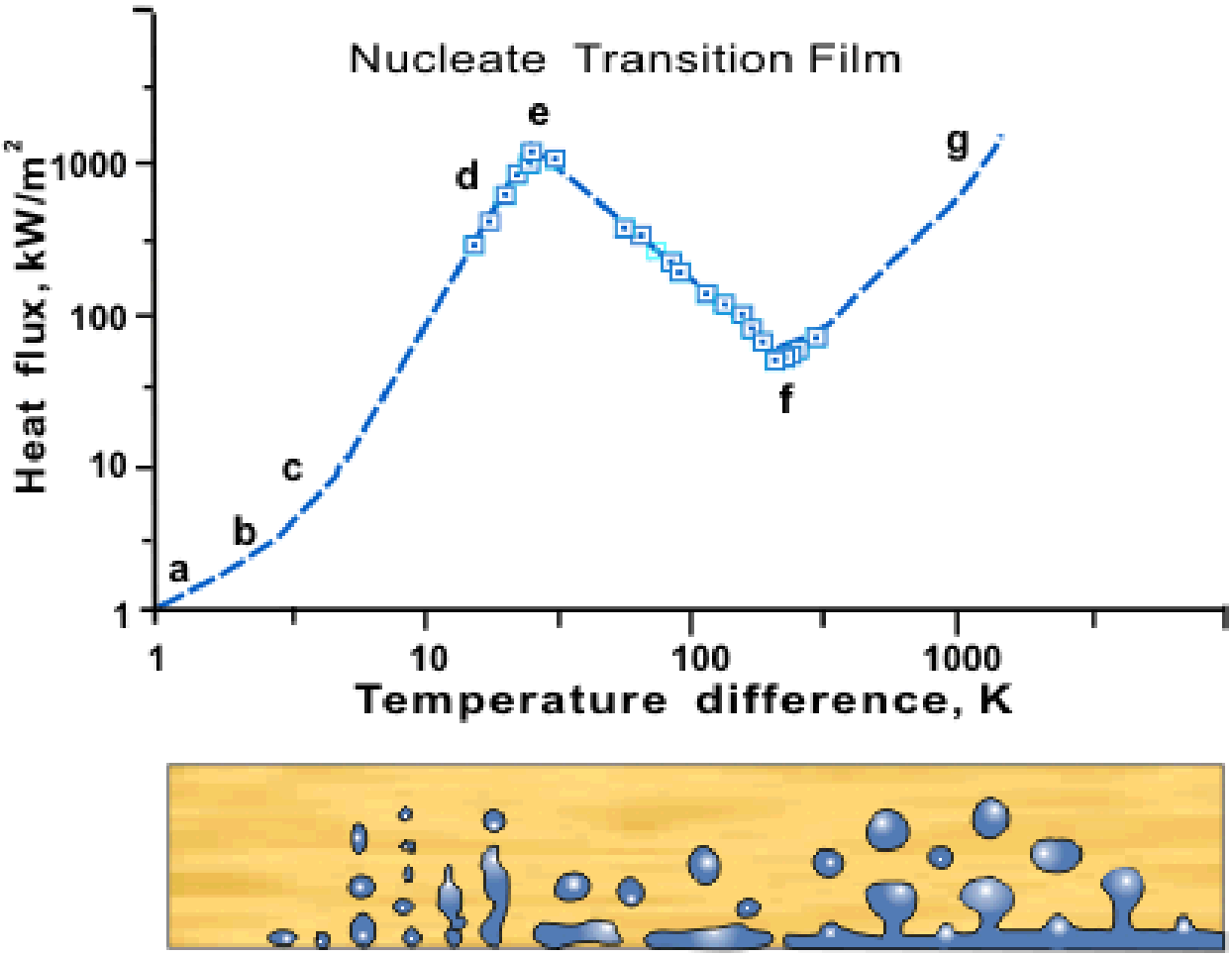


Figure 1 A Typical Boiling Curve [Ref: Winterton, Richard, H.S., Thermopedia]

1.1 Nucleate Boiling

Point b in the Figure 1 corresponds to onset of nucleate boiling, and once nucleate boiling is initiated any increase in surface temperature causes an increase in wall heat flux. With an increase in wall superheat more and more sites participate in nucleation thus setting the stage for isolated bubble regime commonly called as partial nucleate boiling. Transient conduction into liquid adjacent to the surface is the most important mechanism for heat removal in this regime. Upon increasing the wall superheat further, the region c-e of the boiling curve is fully developed nucleate boiling where evaporation is the dominant mode of heat transfer. The early models of nucleate boiling were based on bubble growth and enhanced convection transport in the liquid at the vicinity of the bubble. Apart from empirical correlations, some mechanism based correlations were also proposed. One such correlation obtained by accounting the contribution of transient conduction, microlayer evaporation and natural convection and expressed as:

$$q'' = \frac{K^2}{2}\sqrt{\pi\left(k\rho C_p\right)_l} f D_d^2 N_a \Delta T_w + \left(1 - \frac{K^2 N_a \pi D_d^2}{4}\right)\overline{h}_{nc}\Delta T_w + \overline{h}_{ev}\Delta T N_a \frac{\pi}{4} D_d^2 \tag{1}$$

In the above correlation, the constant, K depends on the ratio of the area of influence of a bubble to the cross sectional area of the bubble, Na is the number of nucleate site density, D_d is the bubble diameter and f is the bubble release frequency, the first two terms on the right side were also used by Mikic and Rohsenow (1961) while the last term was suggested by Judd and Hwang (1976) accounting for the microlayer heat transfer. The value of K proposed by them was √1.8 with the model predictions matching the data as shown in Figure 2. Rohsenow (1962) proposed one of first nucleate pool boiling correlations given in the following form:

$$q'' = \mu_l h_{lv} \left[\frac{g(\rho_l - \rho_v)}{\sigma} \right]^{1/2} \mathrm{Pr}^{-s/r} \left[\frac{1}{C_{sf}} \right]^{1/r} \left[\frac{C_{pl}(T_w - T_{sat})}{h_{lv}} \right]^{1/r} \quad (2)$$

Gaertner (1965) presented a photographic study of the pool boiling of saturated water at atmospheric pressure. The nucleate boiling region was subdivided into four regions depending on the mode of vapor generation. At fully developed nucleate boiling, large billowing clouds of vapor were formed at the surface. He observed that massive bubbles were attached to the surface by numerous vapor stems and described them as vapor mushrooms. These vapor mushrooms were planted in the thermal layer and nourished at the vapor liquid interface by evaporation. The length and diameter of the vapor stems reported by him was found to be inversely proportional to the square root of the heat flux.

Lida & Kobayasi (1969) studied local void fractions averaged with time on horizontal heating surface in a saturated pool boiling of water at atmospheric pressure. For the minimum heat flux point they recorded wall void fraction of 96% implying that liquid was still in contact with the heating surface. Void fractions varied according to the superheat and heat flux of the surface. The map of void fraction described from the measured values indicates the statistical situation of the vapor bubbles in nucleate boiling.

Chekanov (1977) performed the earliest experiments to investigate the interaction between two artificial nucleation sites. Chekanov postulated that the bubbles effect one another by acoustic actions and hydrodynamic mixing. In addition to this, it was reported that when the ratio of cavity spacing to bubble departure diameter (S/D) was less than three, the formation of a bubble at one nucleation site inhibited the activation of another site. On the contrary, when S/D > 3 the formation

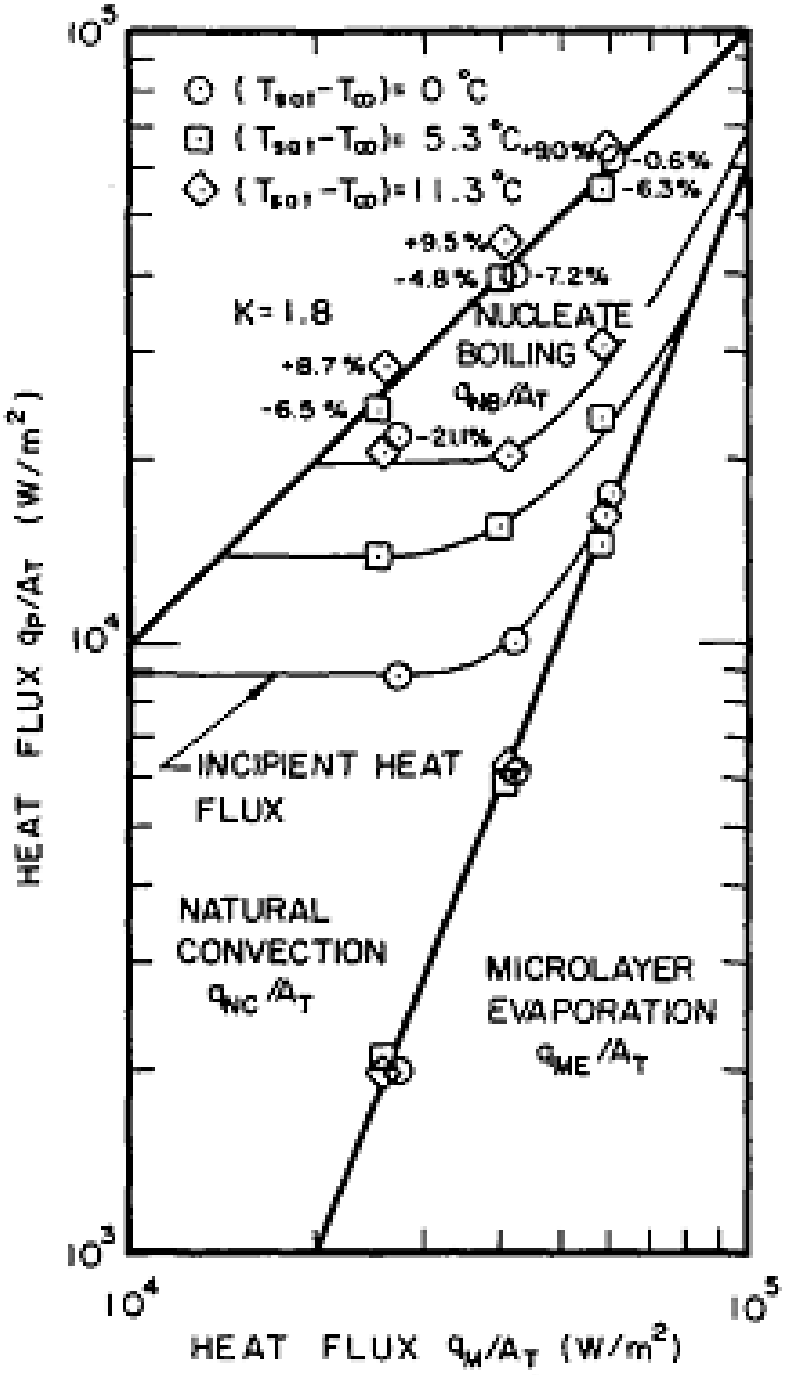


Figure 2 - Nucleate boiling heat transfer model by Judd and Hwang (1976)

of bubble promoted the nucleation of another site. Lastly, if the S/D ratio is very large then there were no interactions between the sites.

Stephan and Abdelsalam (1980) developed a more comprehensive correlation obtained from approximately 5000 data points for nucleate boiling on horizontal surfaces and given as:

$$q" = 0.023\left(\frac{k_l \Delta T}{D_d}\right)\left(\frac{q" D_d}{k_l T_{sat}}\right)^{0.674}\left(\frac{\rho_v}{\rho_l}\right)^{0.297}\left(\frac{h_{lv} D_d^2}{\alpha_l^2}\right)^{0.371}\left(\frac{\rho_l - \rho_v}{\rho_l}\right)^{-1.73}\left(\frac{\rho_l \alpha_l^2}{\sigma D_d}\right)^{0.35} \quad (3)$$

Cooper (1984) proposed a correlation applicable for saturated pool nucleate boiling employing reduced pressure, molecular weight and surface roughness, which is expressed as:

$$q" = 55.0\left[\left\{\left(\frac{P}{P_c}\right)0.12 - 0.21\log_{10}^{R_p}\right\}\left\{-\log_{10}\left(\frac{P}{P_c}\right)\right\}^{-0.55} M^{-0.5}\Delta T_w\right]^3 \quad (4)$$

The above two correlations are among most widely used to predict nucleate boiling heat flux but still have limited applicability as development of comprehensive models are hindered by inability to predict accurately other factors like nucleation sites, bubble frequency and departure diameter, etc.

Lee and Nydahl (1989) performed numerical calculation of bubble growth in nucleate boiling regime using numerical mapping technique. Simulation was done for water at 1 atm and 8.5 K wall superheat. They reported that microlayer was the dominant mechanism for heat transfer providing as much as 90% of the energy for bubble growth and 87% of the overall heat transfer. They also found from their analysis that microconvection heat transfer effects during bubble growth being non-existent.

Liaw and Dhir (1989) performed experiments in a vertical wall using saturated water at atmospheric pressure. The wettability of the surface was controlled by the degree of oxidation of the surface and static contact angle was used as a reference. They inferred that the boiling phenomenon on the heater surface can be subdivided into three regions: the wall dominated region, vapor flow dynamics dominated region, and the intermediate region. They noted that the maximum void fraction occurs at a distance of 1-1.3 mm away from the wall and vapor flow dynamics dominate the region beyond which the maximum void fraction location. The intermediate region is influenced by the wall as well as vapor flow dynamics away from the wall. Their experiments showed that the maximum heat flux values increased as the surface wettability improves. Also, the wall void fraction at which the maximum heat flux occurs is reduced as the contact angle decreases. They further showed that maximum heat flux condition is limited by vapor outflow dynamics for well wetted surface and evaporation rate near the surface for poorly wetted surface.

Wang and Dhir (1993) conducted experiments of pool boiling of saturated water at 1 atm on three surfaces with contact angles of 18°, 35° and 90°. They determined that the active nucleation site density varies as q^2 ,where q is the wall heat flux, and decreases as the surface wettability improves. They found that the spatial distribution and distribution of nearest neighbor distance follows Poisson distribution.

Son *et al.* (1999) performed complete numerical simulation of the hydrodynamics and heat transfer associated with a single bubble using level set method. The computational domain was divided in macro and micro region and hence treated separately with the effect of microlayer being incorporated as the source term in the continuity equation of the macrolayer solution. The

contribution of microlayer to the total heat flux was found to be about 20%. The variation of bubble size and growth with contact angle and wall superheat was also reported.

Basu *et al.* (2002) developed correlations for nucleation site density, independent of flow rate and liquid subcooling but dependent on contact angle. They found that the nucleation site density depends on static contact angle and wall superheat. The effect of velocity, subcooling is implicit in the relation between heat flux and wall superheat and doesn`t influence the nucleation site density independently.

Theofanous *et al.* (2002) presented experimental results of nucleate pool boiling conducted using electrically heated, vapor-deposited sub-micron Ti of 140 nm thickness. Heater`s surface nanomorphology and chemistry were characterized using atomic force microscopy, scanning electron microscopy and X ray diffraction spectroscopy. From the onset of nucleation to the boiling crisis, dynamic thermal patterns on the heater surface were visualized using high speed, high resolution infra red camera. Experimental data of nucleation and boiling heat transfer at high heat fluxes between aged and fresh heaters are presented. They reported a linear relation between q and ΔT for aged heaters which is in stark contrast that the dependence may be as high as third power. As the bubble base corresponds to dark spots on the thermal pattern, it was argued that active sites are effective heat sinks.

Mukherjee and Dhir (2004) numerically studied lateral bubble merger for different orientations, contact angles and superheats for a horizontal surface. Their code was however based on finite volume approach with SIMPLE method to solve the pressure equation and power law scheme to solve the convective terms in the conservation equations. Their results show that bubble merger significantly increases the wall heat transfer due to formation of vapor bridges with

trapping of liquid layer underneath the bubbles and by drawing cooler liquid towards the wall after merging.

Aparajith *et al.* (2006) performed the dynamics and simulation of multiple bubble merger during pool boiling under reduced gravity using PF5060 as the test fluid. Scaling relations for PF5060 with respect to gravity for bubble departure diameter and growth period were reported. The effect of cavity spacing and orientation of cavities on bubble departure diameter, bubble growth period, vapor interfacial structure and heat flux were also investigated.

Li and Dhir (2007) conducted numerical simulations of sliding bubble on a downward facing heater surface. The bubble shape was found to change from spheroids to ellipsoids and finally to a bubble cap. They found that the temperature gradient on the wall behind the bubble is significantly increased by the sliding motion of the bubble leading to enhanced heat transfer on the heater surface.

Sussman *et al.* (2007) presented a coupled level set and volume of fluid method based on sharp interface method for computing two-phase flows of immiscible fluids. Their method can be reduced to a single phase approach in the limiting case of zero gas density and zero gas viscosity. A cell-centered semi-implicit treatment for the viscous terms was developed and hence viscous jump was treated as sharp. Their improved accuracy over conventional continuum approach and ghost fluid method can reliably handle complex interfacial geometries.

Wu *et al.* (2007) coupled the level set method with moving mesh method to simulate subcooled nucleate pool boiling under strong temperature gradient near the surface. Comparison of dimensionless terminal velocities of rising bubbles and growth rate of bubbles for various grid sizes with wall superheat and liquid subcooling was done with improved accuracy due to moving

mesh method.

Son and Dhir (2008) studied numerically nucleate boiling at high heat fluxes on horizontal surfaces with two-dimensional assumption. The effect of contact angle, wall superheat and waiting period on the bubble dynamics and wall heat flux were investigated. The model was used in conjunction with correlations specifying nucleation site density and bubble waiting time. Heat fluxes obtained from the model were within ± 25% of Stephan and Abdelsalam (1980) correlation.

Gerardi *et al.* (2010) used synchronized high speed imaging and infrared thermometry to obtain spatial and temporal resolved information on bubble nucleation and heat transfer in pool boiling of water. Measurements were made for bubble departure diameter, frequency, growth, waiting times as well as for microlayer radius and dryout radius. However, they stated that the relative importance of wall/microlayer heat transfer and superheated liquid layer around the bubble depends on fluid, heater surface characteristics like roughness, wettability, etc and heat flux and couldn`t draw any definitive conclusion.

Golobic *et al.* (2012) examined the growth of an isolated bubble and coalescence between bubbles of dissimilar sizes during nucleate pool boiling of water on horizontal surface. Transient distributions of wall heat flux were mapped using wall temperature measurements made by high speed IR camera.They observed that following contact between the bubble, complex disturbances to the heat flux in the contact areas of both bubbles were seen. The bubble shape oscillations observed during coalescence were more as the difference in the bubbles sizes increases.

Sato and Niceno (2013) used a conservative two phase flow model and introduced a color function as the volume fraction of liquid inside a control volume. In order to maintain strict mass conservation and accurate interface shape, the CIP-CSL2 scheme was used. The smearing of

the interface was further prevented using an interface sharpening equation. They calculated the mass transfer rate directly from the heat flux at the liquid-vapor interface. A three-dimensional verification case of a growing vapor bubble in a superheated liquid has been simulated for zero gravity conditions.

Sielaff *et al.* (2015) performed experimental and numerical studies on horizontal bubble coalescence under varying pressure conditions. The fluid used was FC72 for pressure values of 500,700 and 900 mbar. At low pressures residual droplet was observed for both experimental and numerical work. The frequency of bubble coalescence showed a strong dependence on pressure. For a given nucleation site density distance, an optimal pressure leading to maximum coalescence probability and frequency was demonstrated. While heat transfer rates between experiments and numerical simulations were in agreement, however the bubble dynamics in terms of its shape after merging of the bubbles differed from the experiments.

Gong and Cheng (2015) used the phase change Lattice Boltzmann method and investigated the effect of wettability on saturated pool boiling heat transfer for the case of constant wall superheat on a substrate of finite thickness. Both hydrophobic and hydrophilic surfaces were considered and they reported that a hydrophobic surface has a higher heat flux than a hydrophilic surface at low superheats. Also, for hydrophobic surface a residual bubble was left in comparison to hydrophilic surface and exhibits lower onset of boiling temperature, higher boiling heat transfer at low superheats and lower critical heat flux comparatively. This model however has a limitation on maximum density ratio of approximately 100 which can be simulated.

1.2 Maximum Heat Flux

Maximum heat flux (CHF) represents the upper limit of fully developed nucleate boiling regime (point e of Figure 1). The early models for prediction of maximum heat flux, also known as "far-field model", were proposed by Kutateladze (1948) and Zuber (1959) which are based on the hydrodynamics of vapor flow. Zuber`s hydrodynamic theory assumes that at CHF the vapor escape velocity and vapor flow area fraction reaches their critical values. At the critical vapor velocity the jets in a countercurrent situation become unstable and thus impediment the outflow of vapor. This instability occurs away from the heater surface and is unaffected by the surface conditions. Zuber`s expression for CHF is:

$$\dot{q}_{\max,zub} = \frac{\pi}{24}\rho_v h_{fg}\sqrt[4]{\frac{\sigma g(\rho_l-\rho_v)}{\rho_v^2}}\left[\left(\frac{\rho_l(16-\pi)}{\rho_l(16-\pi)+\rho_v\pi}\right)\left(\frac{\rho_l+\rho_v}{\rho_l}\right)^{1/2}\right] \tag{5}$$

The same theory has been extended by Lienhard & Dhir (1973) to evaluate CHF for pool boiling on finite size heaters. According to this extended hydrodynamic theory, the CHF can be expressed as:

$$\frac{\dot{q}_{\max}}{\dot{q}_{\max,zub}} = f(l') \tag{6}$$

Where,

$$l' = \frac{l}{\sqrt{g(\rho_l-\rho_v)/\sigma}}$$

An alternative model known as "near-surface model" which was proposed by Haramura and Katto (1983) hypothesizes that vapor stems supporting the mushroom type bubbles is

subjected to Helmholtz instability instead. The model also postulates that the macrolayer formed on the surface reduces in thickness while a vapor mass hovers over it. Other factors which influence the CHF are surface contamination, heater thickness and material, system pressure, gravity, mode of heating surface, liquid subcooling and flow velocity (Dhir, 1998).

Ha and No (1998) proposed a dry-spot model for high heat flux nucleate boiling and critical heat flux (CHF) based on Poisson distribution of active nucleation sites. They postulated that a dry-spot is formed when the number of bubbles surrounding one bubble exceeds the critical number thus inhibiting the flow of liquid under the bubble base thus triggering CHF condition. Their model was validated with experimental data of pool boiling including CHF.

Rule and Kim (1999) made detailed local measurements of wall heat flux for saturated pool boiling of FC-72 using an array of 96 heaters in a temperature controlled mode. They obtained the data in nucleate boiling, critical heat flux and transition boiling regimes. The CHF values obtained by them were approximately 33% higher than Zuber`s model. They further observed that the inner heaters reach CHF at lower wall superheats than the array averaged heat flux whereas the maximum heat flux for the edge heaters was observed to be more than the CHF of the inner heaters. In the transition boiling regime, heat transfer during liquid contact was observed to decrease with increase in wall superheat, contrary to the previously reported models.

He *et al.* (2001) proposed a numerical simulation model based on macrolayer model where the thickness of the macrolayer is determined along with the growth rate of vapor stem. By using bisection method, the boiling curve of water and FC-72 were obtained. They concluded that evaporation at the interface of vapor stem-liquid is the main part of the total heat flux. With increasing heat flux, the evaporation due to the decay of macrolayer plays a more important role.

Their boiling curve shifted towards left with the increase of surface roughness with the CHF value remaining almost the same.

Kandlikar (2001) developed a theoretical model for CHF prediction in saturated pool boiling. An important factor in this model was the horizontal component of the force due to the evaporation from the liquid vapor interface. This force exceeds the retaining forces due to surface tension and gravity resulting the spreading of the vapor bubble along the heater surface and thus initiating the CHF condition. The proposed model considers the effect of dynamic receding contact angle and subcooling and predicts the experimental data very well for water, refrigerants and cryogenic liquids. The model considered the range of contact angle from 20° to 110° and the predicted CHF value decreases as the contact angle increases.

Theofanous *et al.* (2002) conducted burnout experiments on fresh and aged heaters with CHF`s varying from 50% to 140% of the hydrodynamic limit. Based on the high speed, high resolution infrared thermometry, they concluded that CHF cannot be (macro) hydrodynamically limited rather it is controlled by the microhydrodynamics and rupture of an extended liquid microlayer, They remarked that dynamics and stability of the liquid microlayer were likely to be governed by the distance between the neighboring nucleation sites. At high superheats, both reversible and irreversible dry spots were observed by them. Through the reversible dryspots only irreversible dry spots arise. The dynamics of the reversible dry spot is affected by nearby nucleation event while an irreversible dry spot is characterized by the growth velocity rate of millimeters per second just prior to thermal runway.

Zhao *et al.* (2002) proposed a model to predict the nucleate boiling regime including the CHF. Their model gives a dynamic structure of vapor liquid-solid contacts. In their model heat

transfer is mainly attributed to the evaporation of the microlayer formed periodically during the inception of new bubbles. The local evaporation and partial dryout speed of the microlayer increases with increase in wall superheat due to thinning of the microlayer. According to them, the developing process of the isolated dry-areas beneath individual bubbles results in CHF.

Auracher *et al.* (2004) performed experiments for steady state boiling curve in the transition boiling regime with a clean heater surface and didn`t observe any hysteresis. They also presented a concept of reaction-diffusion model to predict CHF where instability of the dry spots on the surface is accompanied by a temperature wave. The CHF values obtained by them were higher than those given by the hydrodynamic theory.

Luttich *et al.* (2004) presented a unifying correlation to calculate the heat flux along the entire boiling curve. They supplemented their model by relying on experimental observations to calculate the interfacial geometry close to the boiling surface. Key parameters used in their model were vapor fraction, interfacial velocity, interfacial line (contact line), area densities and their respective fluxes on the boiling surface. They successfully correlated the boiling curve of FC 72 and CHF data for water with interfacial geometry found using optical probe measurements and applying multiphase flow averaging theory.

Das *et al.* (2006) developed an analytical model of heat transfer applicable for pool boiling based on the evaporation of micro and macrolayers during the vapor bubble growth. They ignored the influence of adjacent bubbles on a particular growing bubble which is not true for the case of high heat fluxes. The effect of fluid motion was also ignored. The boiling curve was obtained by considering the bubble dynamics and decreasing thickness of liquid layer along with the increase in the dry spot radius.

Chu and Yu (2009) developed a new comprehensive model for nucleate pool boiling of pure liquid upto CHF. The model proposed by them was expressed in terms of total number, minimum and maximum sizes of active nucleation sites, fractal dimension, wall superheat and properties of fluids. The model was shown to be in good agreement with the experimental data.

Guan *et al.* (2011) proposed a new mechanistic model for predicting CHF in pool boiling system with pentane, hexane and FC-72.They postulated that the film boiling occurs when the vapor momentum flux is sufficiently high to lift the liquid macrolayer from the surface thus inhibiting wetting. Their obtained expression for CHF is of the same form as Zuber`s instability model but over predicts the CHF value obtained from Zuber`s model for the case of higher pressures at 300-450 kPa.

Buchholz *et al.* (2006) used an array of 36 microthermocouples embedded in a horizontal copper heater. The test fluids used for their study was isopropanol and FC-3284. They observed very localized and rapid temperature drops at the bottom of the bubbles in the nucleate boiling. At low heat flux nucleate boiling the vapor is slightly superheated while the liquid superheat is strong. However, at high wall superheats the vapor becomes even more superheated while the liquid is close to saturation temperature. A liquid rich layer, whose thickness decreases with increasing superheat, above the surface was confirmed by micro optical probe in their experiments. At CHF conditions, dry patches were observed at the surface while at the film boiling surface temperature fluctuations were very weak. At film boiling, the vapor superheat present in the bubble decreases with increasing distance.

1.3 Transition Boiling

Region e-f of the Figure 1 shows the transition boiling regime characterized by reduction in surface heat flux with an increase in wall superheat and can be bypassed if the boiling curve is performed in a heat flux controlled setting. It has traditionally been interpreted as a combination of nucleate and film boiling alternatively and is probably the least understood regime as it is practically of less importance than nucleate boiling along with the fact that its mechanisms are far more complicated. Since transition boiling can be viewed as a combination of liquid and solid contacts, Berenson (1962) expressed the transition boiling heat flux as:

$$q'' = Fq_l'' + (1-F)q_v'' \tag{7}$$

where,

$$F = \exp\left(-2.2\frac{\Delta T}{\Delta T_{CHF}} + 2\right)$$

In the above expression, F is the average ratio of the liquid contact area at any given instance, ΔT_{CHF} is the wall superheat at critical heat flux (CHF), q''_{max}. With the assumption that a unique transition boiling curve exists, another correlation was given by Kalinin *et al.* (1976) as:

$$q'' = q_l'' F_l + \left(1 - \mathrm{F_l}\right) \mathrm{q_v''} \tag{8}$$

Where, q_l and q_v are the heat fluxes associated with liquid and vapor respectively.

Lee *et al.* (1985) did the measurement of the local surface temperature history in transition and film boiling on surfaces at high wall superheats. From their measurements they observed direct liquid solid contacts for both the transition and film boiling regimes. Both the local liquid

contact fraction and the average contact duration increased monotonically with decreasing surface superheat. They inferred that the liquid-solid contacts might be the dominant mechanism of energy transport in the transition boiling process.

Ramilson and Lienhard (1987) re-created the Berenson flat plate transition boiling experiment and conducted experiments with several fluids on mirror-polished, roughened and teflon coated surface. They confirmed the prior conclusion that the burn out heat flux suffers a secondary dependence on surface conditions. They also proposed a model to calculate the heat flux at the boundary between pure-film and film transition regime as a function of the advancing contact angle.

Dhir and Liaw (1989) developed a framework for theoretical prediction of boiling curve except in the isolated bubble regime. By assuming the existence of stationary vapor stems at the wall two dimensional steady state conduction equations are solved to determine the temperature distribution and the heat transfer into the thermal layer. By employing experimentally observed void fractions, apart from nucleate and transition boiling heat fluxes, maximum and minimum heat fluxes with their dependence on contact angle are also predicted from their model . Additionally, the model predicted values of vapor stem spacing and diameter to decrease with increase in temperature and contact angle.

Maruyama *et al.* (1992) simulated the transition boiling curve by developing an instantaneous heat flux model. Their heat flux model was based on the combined effect of increase of void fraction and the decay of the macrolayer thickness with time. Spatial pattern of the vapor stems with random positions and sizes was assumed initially. An initial thickness of the macrolayer was taken from Rajvanshi *et al.* (1990) and the time varying model of the macrolayer

thickness at any time could be solved independently of the void fraction.

1.4 Film Boiling

At point f of the boiling curve, due to very high wall superheats a layer of vapor film completely blankets the surface and transport of heat across the vapor film from the wall is achieved by conduction, convection and radiation. Point f demarcates the boundary between transition boiling and film boiling regime and corresponds to the minimum heat flux condition of the boiling curve. It is alternatively referred as the Leidenfrost point.

Berenson (1961) obtained the expression for the heat transfer coefficient during saturated film boiling on horizontal surfaces, given in the form of Nusselt number (Nu) as:

$$Nu = 0.42\left[\frac{\rho_v\left(\rho_l-\rho_v\right)gh_{fg}}{k_v\mu_v\Delta T}\right]^{1/4}\left[\frac{\sigma}{g\left(\rho_l-\rho_v\right)}\right]^{3/8} \tag{9}$$

Klimenko (1981) carried out generalized analysis of film boiling on horizontal flat plate and developed correlations for variety of liquids including cryogen.

$$\begin{aligned} Nu &= 3.02\times10^{-2}\,\mathrm{Ar}^{1/3}\,\mathrm{Pr}^{1/3}\,f_1(\beta) \quad \text{for } \mathrm{Ar} < 10^8 \\ &= 1.37\times10^{-3}\,\mathrm{Ar}^{1/2}\,\mathrm{Pr}^{1/3}\,f_2(\beta) \quad \text{for } \mathrm{Ar} < 10^8 \end{aligned}$$

where,

$$\begin{aligned} f_1 &= 1 \quad \text{for } \beta > 0.71 & f_2 &= 1 \quad \text{for } \beta > 0.50 \\ &= 0.71\beta^{-1/2} \quad \text{for } \beta < 0.50 & &= 0.89\beta^{-1/3} \quad \text{for } \beta < 0.71 \end{aligned} \tag{10}$$

In the above expression, Ar is the Archimedes number $\left((2\pi)^3 \, g l_o^{\,3} \rho_v (\rho_l - \rho_v) \Big/ \mu_v^2 \right)$ and β is the ratio of sensible heat to latent heat.

Juric and Tryggvason (1998) presented front tracking method to simulate liquid vapor phase change using a single field formulation with just one set of conservation equations of mass, momentum and energy. The interface matching conditions are implicitly imposed by the interfacial source terms with the help of delta functions which leads to smeared out profiles for different variables. Two dimensional film boiling simulations were done and overall heat transfer rates and wall temperatures were compared with experimental observations and correlations.

Esmaeeli and Tryggvason (2004) performed direct numerical simulation of film boiling using one set of conservation equations to represent heat transfer, mass transfer, and fluid flow for both the liquid and the vapor phase. The method described by them had also been extended to study multimode film boiling (where evolution of phase boundary leads to formation of bubbles of different sizes and spacings) on horizontal surfaces, explosive boiling and boiling in complex geometries. By imposing periodic boundary conditions at the horizontal boundaries, evolution of liquid/vapor interface and velocity field during film boiling process at Ja = 0.064 was shown.

Tomar *et al.* (2005) presented a coupled level set and volume of fluid approach for modeling incompressible flows with surface tension. Planar simulations of bubble growth in water at near critical pressure and R134a refrigerant was done. The effect of saturation pressure on the bubble formation frequency was also studied. At near critical pressures, for film boiling, there exists a critical superheat beyond which jet-like-columns arise at the nodes while bubble growth and detachment occurs periodically at the anti-nodes. They also did the Fast Fourier Transform

(FFT) of the space-averaged Nusselt number and reported two dominant frequency peaks for a superheat of 10 K.

Gibou *et al.* (2006) described a sharp interface capturing method by using a ghost fluid approach to impose the jump conditions at the interface. They applied their algorithm for the simulation of two dimensional film boiling. Their treatment preserves the discontinuous nature of all variables across the interface except for the viscosity, for which a delta formulation mechanism was used for simplicity. Furthermore, viscosity was treated as explicit. The jump conditions in the pressure derivatives were balanced by considering real velocities.

From the above studies it is concluded that although several theoretical, experimental and numerical work has been done yet no study has been focused on a unified approach to numerically predict the entire boiling curve. The importance of nucleation site density along with the contact angle is vital in predicting the heat fluxes at the surface. Several attempts were made to predict the heat flux for a particular regime but they didn`t account the combined effect of nucleation site density and the waiting time for a specified contact angle.

1.5 Objective of the Present Study

The objective of the present study is to numerically reproduce the boiling curve in a temperature controlled mode using a unified numerical model supplemented by correlations specifying nucleation site density and bubble waiting time as a function of wall superheat for different contact angles. Steady state boiling curve for all the three regimes *viz.* nucleate, transition and film boiling has been obtained with a unified numerical model by incrementing the wall superheat for a range of static contact angles for both two dimensional and three dimensional cases. Additionally, a 3D parallel framework has also been implemented to enhance the performance of the code with the future intention of optimization and scaling up.

2. Numerical Model

Some of the methods used to study multiphase process numerically are:

1. Marker and Cell Method (MAC) : Developed by Harlow and Welch (1965) , it was one of the earliest attempts made to compute solution of fluid dynamics problems with free boundaries were made in 1960s. The interface is marked by weightless particles which under the influence of velocity field are convected and can be explicitly used to reconstruct the interface on a fixed mesh.
2. Volume of Fluid Method (VOF) : Developed in 1970s the Volume of Fluid Method (VOF) employs a piecewise constant scalar field to track the location of both the phases. Information about the volume fraction of one of the phases in a cell is contained and convected under the influence of velocity field. This implicit approach allows simple handling of topological changes but unlike MAC schemes does not involve any additional consideration while explicit reconstruction of the interface.
3. Arbitrary Lagrangian-Eulerian Method (ALE): ALE method is the finite element method in which the computational domain is not fixed a priori. The mesh follows the interface and can precisely track the motion of the interface. It helps to combine the respective advantages of the Lagrangian and Eulerian formulations however its implementation requires the formulation of mesh-update procedure that assigns mesh-node velocities at each time step.
4. Level-Set Method (LS) : Introduced at the end of 1980s by Osher and Sethian (1988), the interface is defined as the zero level set of the continuous higher dimensional scalar field. Interface and shapes are tracked by a level set function which is convected in the velocity

field. Similar to the VOF the LS technique reduces the complexity in handling the interface during topological changes like breaking and merging , but in contrast the level set function is globally continuous and hence the solution is less effected by numerical diffusion during the transport process.

5. Lattice Boltzmann Method (LB): LBM for simulating immiscible binary fluids in two dimension were introduced in 1990s by Gunstensen *et al.* (1991). The liquid or vapor state at each lattice point is determined by the thermodynamic relations given by equation of state thus eliminating the need to track the interface explicitly.

2.1 Governing Equations

Direct numerical simulation (DNS) of the two phase flows with application to boiling heat transfer has been performed since 1990s. Following the work of Son *et al.* (1999) and Son and Dhir (2008), the computational domain is divided into macrolayer and microlayer (Figure 3) . The microlayer is discussed in Section 2.7 while for the macrolayer, the governing equations for global conservation of mass, momentum and energy for each phase are given as:

$$\nabla . u_f = 0$$

$$\rho_f \left(\frac{\partial u_f}{\partial t} + u_f . \nabla u_f \right) = -\left(\nabla p \right)_f + \rho_f g + \nabla . \mu_f \left(\nabla u + \nabla u^T \right)_f$$

$$\rho_f c_f \left(\frac{\partial T_f}{\partial t} + u_f . \nabla T_f \right) = \nabla . k_f \left(\nabla T \right)_f \tag{11}$$

The subscript *f* denotes the phase (liquid or vapor). The interface between the different phases is captured using a level set function Φ. The interface, for all times $t \in [0, T]$ is given by

$$\Gamma_f(\mathrm{t}) = \left\{ \vec{x} : \phi(\vec{\mathrm{x}}, \mathrm{t}) = 0 \right\} \tag{12}$$

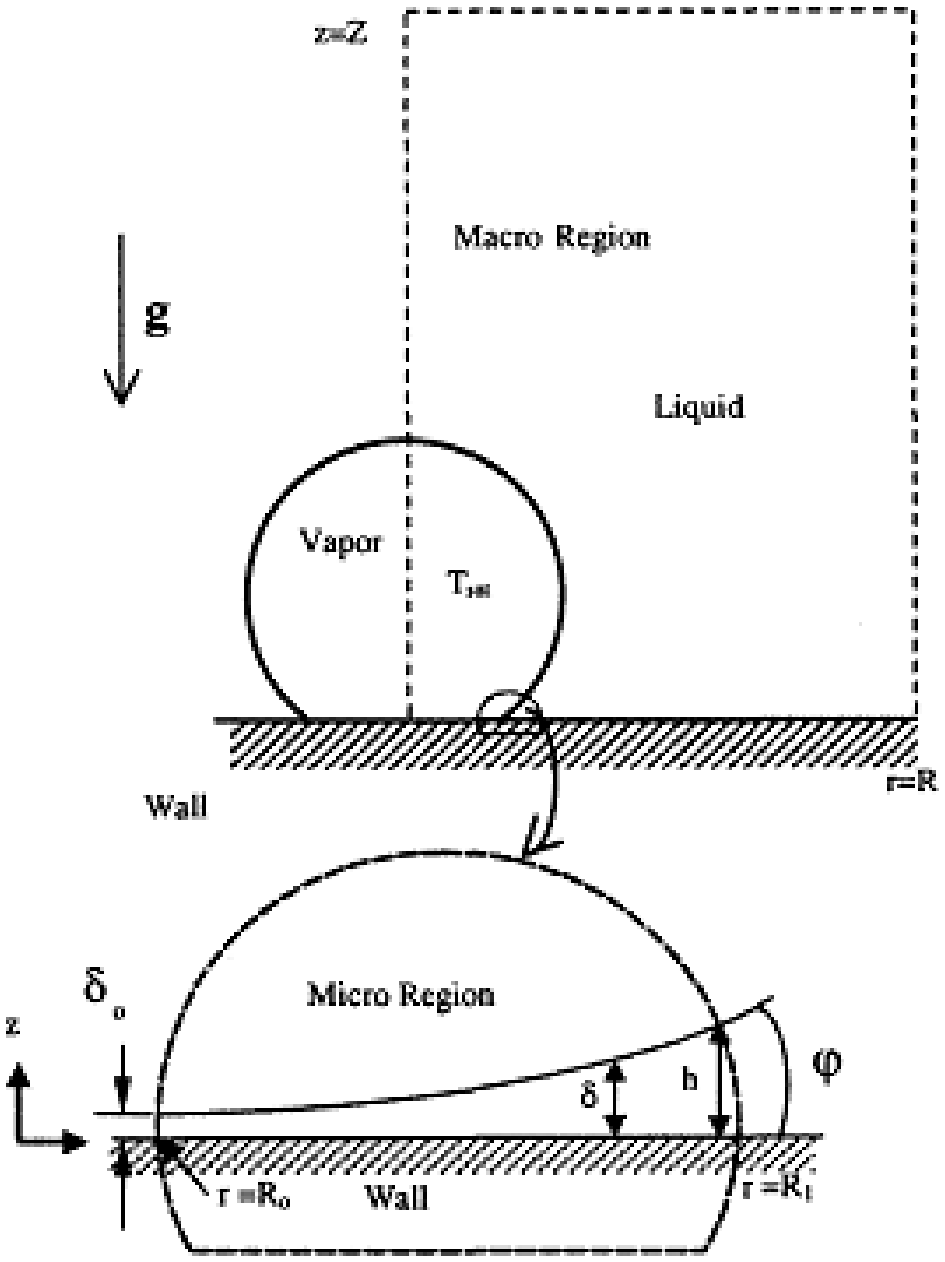


Figure 3 Numerical simulation model with micro region and macro region (Son *et al.* 1999)

The Level-Set Equation (LSE) is typically a smooth (Lipschitz continuous) function defined as:

$$\frac{\partial \phi}{\partial t} + U_{\text{int}} . \nabla \phi = 0 \tag{13}$$

Where U_{int} is defined as:

$$U_{\text{int}} = u_f + \frac{\dot{m}\vec{n}}{\rho_f} \tag{14}$$

ϕ is defined as the signed distance function such that

$$\phi(\vec{x},t) = \begin{cases} < 0 \ if \ \vec{x} \in \Omega_{liq} \\ = 0 \ if \ \vec{x} \in \Gamma_{\text{int}} \\ > 0 \ if \ \vec{x} \in \Omega_{vap} \end{cases} \tag{15}$$

In the above expression, Ω_{liq} and Ω_{vap} are domains of liquid and vapor phase, and Γ_{int} is the interface between them. The solution of level set function does not satisfy the condition of a distance function, i.e., its gradient becomes very large or very small near the interface for which it is reinitialized to a distance function by a smoother, less distorted function by solving the following equation (Sussman *et al.,*1994) :

$$\frac{\partial \Phi}{\partial \tau} = S(\Phi_o)\left(1 - |\nabla \Phi|\right) \tag{16}$$

Where,

$$S(\Phi_o) = \frac{\Phi_o}{\sqrt{\Phi_o + h^2}} \tag{17}$$

Here h is the grid spacing. Since the grid spacing is not constant throughout the domain, effective grid spacing for level set function is defined as (Son and Dhir, 2007):

$$h = \sqrt{\frac{\left(\frac{\partial \Phi}{\partial x}\right)^2 (\Delta x)^2 + \left(\frac{\partial \Phi}{\partial y}\right)^2 (\Delta y)^2 + \left(\frac{\partial \Phi}{\partial z}\right)^2 (\Delta z)^2}{\left(\frac{\partial \Phi}{\partial x}\right)^2 + \left(\frac{\partial \Phi}{\partial y}\right)^2 + \left(\frac{\partial \Phi}{\partial z}\right)^2}} \tag{18}$$

The normal to the interface n and the interface curvature κ are defined as:

$$\vec{n} = \frac{\nabla \Phi}{|\nabla \Phi|}$$

$$\kappa = \nabla . \vec{n} = \nabla . \frac{\nabla \Phi}{|\nabla \Phi|} = \frac{\Phi_{xx}\Phi_y^2 - 2\Phi_x \Phi_y \Phi_{xy} + \Phi_{yy}\Phi_x^2}{\left(\Phi_x^2 + \Phi_y^2\right)^{3/2}} \tag{19}$$

For the treatment of the surface tension acting on the interface, the Continuum Surface Force (CSF) developed by Brackbill *et al.* (1992) is used, where the interfacial tension is transformed into a body force for the cells containing the interface. Other ways to treat surface tension are Continuum Surface Stress (CSS) model and using a variational approach using the Laplace-Beltrami operator on the free surface. The former approach is generally used where variation of surface tension force along the interface is considered.

2.2 Assumptions

1. Material properties like density, viscosity, specific heat and thermal conductivity are taken as constant for each phase.
2. Both the fluids are newtonian and incompressible.
3. Effect of turbulence is negligible.
4. Interface is maintained at the saturation temperature.
5. Uniform wall superheat is applied to the surface.

2.3 Boundary Conditions

The boundary conditions for the governing equations are defined as:

At the wall (y=0) :

$$\begin{aligned} & u = v = 0 \ ; \ \mathrm{w} = 0 \ (\mathrm{For\ 3D}) \\ & T = T_{wall} \\ & \frac{\partial \Phi}{\partial y} = -\cos\varphi \\ & \frac{\partial p}{\partial y} = \rho g \end{aligned} \tag{20}$$

At the planes of symmetry:

$$\begin{aligned} & \text{At x = 0, L:} \quad \frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = \frac{\partial w}{\partial x} = \frac{\partial T}{\partial x} = \frac{\partial \Phi}{\partial x} = \frac{\partial p}{\partial x} = 0 \\ & \text{At z = 0, L:} \quad \frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = \frac{\partial w}{\partial z} = \frac{\partial T}{\partial z} = \frac{\partial \Phi}{\partial z} = \frac{\partial p}{\partial z} = 0 \ \ \text{(For 3D)} \end{aligned} \tag{21}$$

At the top boundary (y=H) :

$$\frac{\partial u}{\partial y} = \frac{\partial v}{\partial y} = \frac{\partial \Phi}{\partial y} = 0 \; ; \; \frac{\partial w}{\partial y} = 0 \text{ (For 3D)}$$
$$p = p_{atm} \tag{22}$$
$$T = T_{sat}$$

The boundary condition of pressure $\left(\frac{\partial p}{\partial y} = \rho g\right)$ at y = 0 is converted into homogeneous Neumann boundary condition $\frac{\partial p}{\partial y} = 0$ by projecting Equation 34 onto the outer unit normal of the domain boundary (Croce *et al,* 2004).

2.4 Interface Velocity and Jump Conditions

Liquid and vapor phases are separated across an interface through which phase change occurs. A numerical approach to the multiphase flow problems requires tracking/capturing the interface as well as enforcing the appropriate boundary conditions at the tracked/captured interface. Typically δ-function formulation is used to enforce the appropriate boundary conditions across the interface which suffers from the drawback that the δ-function formulation smears out the numerical quantities across the interface (Nguyen *et al., 2001*). Hence, to address the aforementioned shortcomings the discretization of the equations near the interface which is not coincident with the grid points is facilitated using ghost fluid method (GFM) as proposed by Fedkiw *et al.* (1999).

Since mass transfer is occurring at the interface normal velocity is discontinuous across the interface. From conservation of mass across the interface:

$$\rho_l \left[u_{l,n} - u_{\text{int,n}} \right] = \rho_v \left[u_{v,n} - u_{\text{int,n}} \right] \tag{23}$$

Where $u_{l,n}, u_{v,n}, u_{\text{int,n}}$ are the liquid, vapor and interface velocities normal to the interface obtained by taking the dot product of the velocity vector and normal vector, which gives the following velocity and pressure jump conditions at the interface provided the interface is not a contact discontinuity (Gibou *et al.* (2006)):

$$\dot{m} = \frac{1}{h_{lv}} \vec{n}.(\mathrm{k_l} \nabla \mathrm{T_l} - \mathrm{k_v} \nabla \mathrm{T_v}) \tag{24}$$

$$u_{l,n} - u_{v,n} = \dot{m} \left(\frac{1}{\rho_v} - \frac{1}{\rho_l} \right) = \dot{m} \alpha \tag{25}$$

$$p_l - p_v = \sigma \kappa - \alpha \dot{m}^2 + \vec{n}.\left[\mu_l \left(\nabla \vec{u} + \nabla \vec{u}^T \right)_l - \mu_v \left(\nabla \vec{u} + \nabla \vec{u}^T \right)_v \right] \tag{26}$$

As explained in detail by Nguyen *et al. (*2001), the tangential velocities and stresses are however continuous but are completely uncoupled for the case of contact discontinuity. By using δ-function formulation, smearing of the jump conditions can occur which, for the case of velocity jump conditions, adds a compressible character to the flow field in the near vicinity of the interface as the divergence free conditions are not exactly satisfied. By implementing the above mentioned jump condition, the conservation equations of both the phases are coupled as (Gibou *et al.* (2006), Son and Dhir (2008)) :

$$\nabla .u = \alpha \dot{m}\vec{n}.\nabla H + \dot{V}_{micro}$$

$$\hat{\rho}\left(\frac{\partial u}{\partial t} + u.\nabla u\right) = -\left(\nabla p + \left(\sigma\kappa - \alpha\dot{m}^2\right)\nabla H\right) + \hat{\rho}\left[1 - \beta_T\left(T_f - T_{sat}\right)\right]g + \nabla .\hat{\mu}\left(\left(\nabla u - \alpha\dot{m}\vec{n}.\nabla H\right) + \left(\nabla u - \alpha\dot{m}\vec{n}.\nabla H\right)^T\right)$$

$$\hat{\rho}\hat{c}\left(\frac{\partial T}{\partial t} + u.\nabla T\right) = \nabla .\hat{k}\left(\nabla T\right)$$

where,

$$\begin{aligned}
H &= 1 \; if \; \Phi > 0 \\
&= 0 \; if \; \Phi \le 0 \\
u_l &= u + \upsilon_{lv}\dot{m}\vec{n}(1 - \mathrm{H}) \\
u_v &= u - \upsilon_{lv}\dot{m}\vec{n}(\mathrm{H}) \\
\hat{\rho} &= \rho_v(1 - \mathrm{F}) + \rho_l F \\
\hat{c} &= c_v(1 - \mathrm{F}) + c_l F \\
\hat{\mu}^{-1} &= \hat{\mu_v}^{-1}(1 - \mathrm{F}) + \hat{\mu_l}^{-1} F \\
\hat{k_f} &= \hat{k_l} = k_l / F \\
F &= 1 \; \; if \; H\left(\phi_A\right) = H\left(\phi_B\right) = 1 \\
&= 0 \; \; if \; H\left(\phi_A\right) = H\left(\phi_B\right) = 0 \\
&= \frac{\max(\phi_A, \phi_B)}{\max(\phi_A, \phi_B) - \min(\phi_A, \phi_B)} \; \; otherwise
\end{aligned} \tag{27}$$

In the above equation ϕ_A, ϕ_B are evaluated at the adjacent grid points and $\dot{V}_{micro}$ is obtained from microlayer solution which is explained in Section 2.7.

2.5 Discretization

For discretization of the governing equations spatially, a staggered grid system is used. This method was first used by Harlow and Welch (1965) in their MAC method for free surface flows as a variant of PIC method. Since the pressure gradient is not calculated *a priori* and is computed during the course of the solution it is possible to create a pressure field with checkerboarded pattern which can persist in the final solution. The remedy for such checkerboarding is staggered grid where fluid and interface velocities are defined at cell faces while pressure (and other scalars) at cell centers, as shown in Figure 4. The density at the cell center is defined as:

$$\rho_P = \rho_v + (\rho_l - \rho_v) H\left(\Phi_P\right) \tag{28}$$

Where

$$\begin{aligned} H\left(\Phi_P\right) &= 1 \quad if \quad \Phi > 0 \\ &= 0 \quad if \quad \Phi \le 0 \end{aligned} \tag{29}$$

The effective viscosity is calculated from the level set function as

$$\begin{aligned} \mu &= \mu_l \quad \text{if } \Phi > 0 \\ &= \mu_v \quad \text{if } \Phi \le 0 \\ &= \frac{\Phi_{\max}\mu_l^{-1} - \Phi_{\max}\mu_v^{-1}}{\Phi_{\max} - \Phi_{\min}} \quad \text{for all other cases.} \\ &\text{where,} \\ \Phi_{\max} &= \max(\Phi_{i,j-1/2}, \Phi_{i-1,j-1/2}) \\ \Phi_{\min} &= \max(\Phi_{i,j-1/2}, \Phi_{i-1,j-1/2}) \end{aligned} \tag{30}$$

For discretization of continuity equation in each phase, ghost fluid methodology is used. As the normal velocity is discontinuous across the interface, care has to taken when applying numerical discretization in the vicinity of the interface. Following GFM a band of ghost cells is populated on either side of the interface that can be used by the fluid velocities on the other side of the interface. The level set function, Φ is defined as the cell center and its offset values at cell faces can be computed with simple averaging. The cell face values can then be used to determine which velocity field corresponds to water or vapor phase.

$$\left(\nabla . u_f\right)_P = \frac{u_{l,e} - u_{l,w}}{\Delta x} + \frac{v_{l,n} - v_{l,s}}{\Delta y} \tag{31}$$

If the problem is convection dominated then the accuracy of the convective term is essential for the spatial discretization. The principle behind ENO interpolation is in using the smoother approximation where smoothness is measured by the absolute value of the second order divided difference (John and Novo, 2012). For discretization of momentum equation spatially the convection terms are thereby treated by second order ENO scheme while the viscous diffusion terms by second order central difference scheme. Treating the viscosity implicitly is preferred but it`s implicit formulation is very difficult (Gibou *et al.,* 2006) hence the semi-implicit treatment of the viscous terms is implemented in which viscosity is explicitly treated and has been shown to be unconditional stable by Li *et al*. (2000). For temporal discretization the convection and source terms are treated by first order explicit scheme and diffusion term by fully implicit scheme.

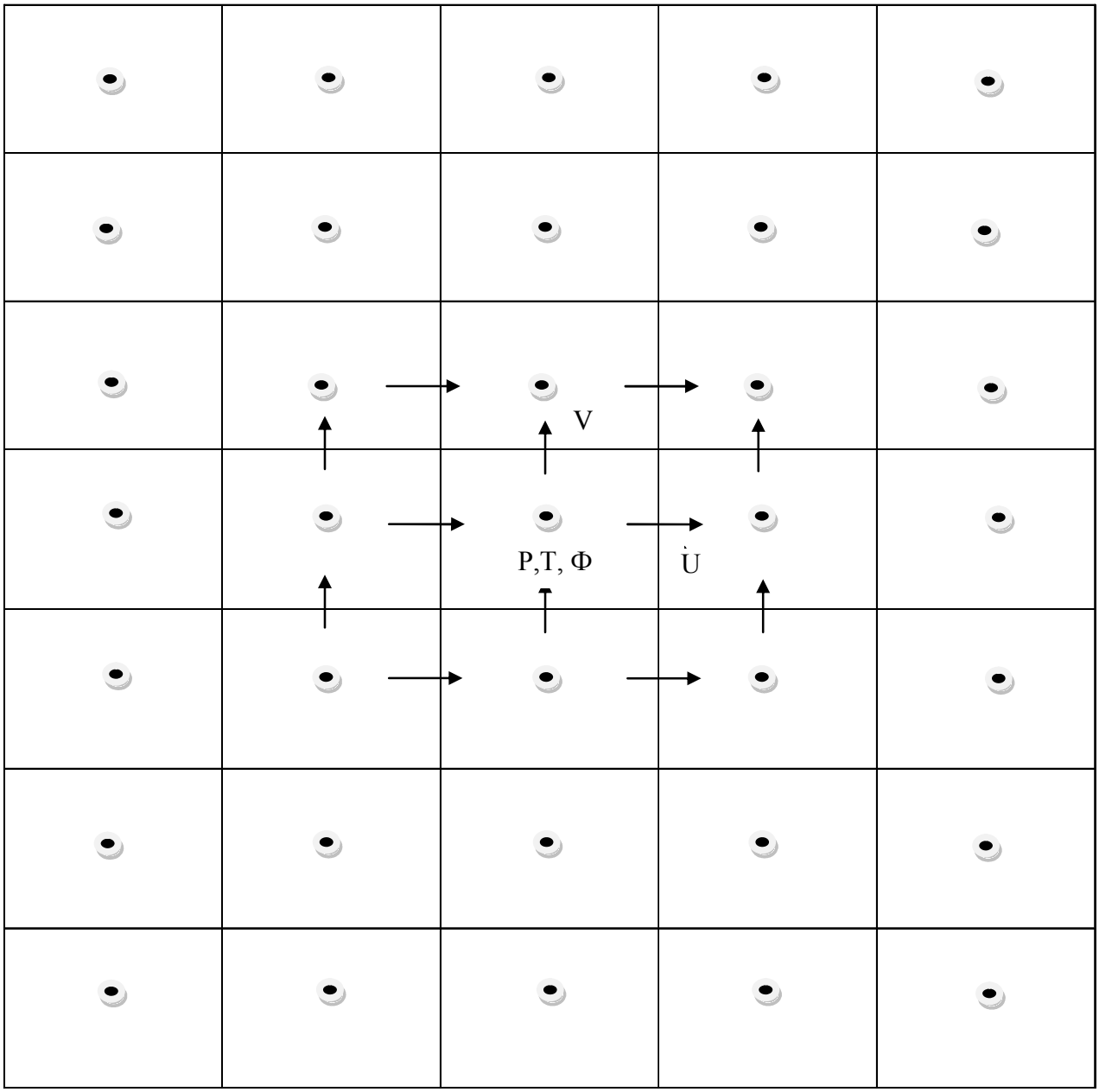


Figure 4 Staggered grid arrangement

$$\frac{u_f^{n+1} - u^*}{\Delta t} = -\frac{1}{\rho_f}(\nabla p)_f + \frac{1}{\rho_f}\left(\nabla.\mu_{eff}\nabla u^{n+1} + S_u^n\right)$$

Where,

$$u^* = u_f^n + \Delta t\left(-u_f^n.\nabla u_f^n + \vec{g}\right)$$

$$S_u = \nabla.\mu_{eff}\nabla u^T - \nabla.\mu_{eff}\left[\left(\alpha\dot{m}n\nabla H\right)^T + \left(\alpha\dot{m}n\nabla H\right)\right] \tag{32}$$

The momentum equation is solved using projection method. The projection method is an efficient way of solving time dependent incompressible Navier-Stokes equations by decoupling velocity with pressure. It consists of three stages, first the intermediate velocity field is computed by solving the momentum equation without the pressure gradient term.

$$\frac{u_f^{**} - u^*}{\Delta t} = \frac{1}{\rho_f}\left(\nabla.\mu_{eff}\nabla u^{**} + \left(\nabla.\mu_{eff}\nabla u^T - \nabla.\mu_{eff}\left[\left(\alpha\dot{m}n\nabla H\right)^T + \left(\alpha\dot{m}n\nabla H\right)\right]\right)^n\right) \tag{33}$$

$$\frac{u^{n+1} - u^{**}}{\Delta t} = \frac{1}{\tilde{\rho}}\left[\nabla p + \left(\sigma\kappa - \alpha\dot{m}^2\right)\nabla H^{n+1}\right] \tag{34}$$

The second stage defines the solution of the following Poisson equation to satisfy the incompressibility condition. For incompressible flows pressure is not a thermodynamic variable and hence cannot be related to equation of state rather it is a Lagrange multiplier which constraints the velocity field to remain divergence free (Gresho and Sani, 1987). Pressure propagates at infinite speed in order to keep the flow incompressible and it is often computationally expensive to compute. The resulting system of linear equations obtained from discretization of the Poisson equation is hence solved using preconditioned conjugate gradient algorithm using Multigrid as the

preconditioner to accelerate convergence.

$$\nabla.\frac{1}{\tilde{\rho}}\nabla p = \frac{\nabla.u^{**} - \left(\alpha\dot{m}\vec{n}.\nabla H\right)^{n+1}}{\Delta t} - \nabla.\frac{1}{\tilde{\rho}}\left(\sigma\kappa - \alpha\dot{m}^2\right)\nabla H^{n+1} \tag{35}$$

The final stage requires the corrected flow field velocity to be projected to the divergence free field as:

$$u_f^{n+1} = u_n^* - \Delta t\frac{\nabla p^{n+1}}{\tilde{\rho}} \tag{36}$$

The spatial discretization of the energy equation is done by implementing the dirichlet boundary condition $T_{int} = T_{sat}$ at the interface (Ishii (1975)). The convection term is discretized using a second order ENO scheme while the diffusion term by fully implicit scheme. The energy equation takes the following form:

$$\frac{T_f^{n+1} - T_f^n}{\Delta t} = -u_f^n.\nabla T_f^n + \frac{1}{\rho_f c_f}\left(\nabla.k\nabla T\right)_f^{n+1} \tag{37}$$

The diffusion term is discretized as (Son and Dhir, 2008):

$$\left(\frac{\partial}{\partial x}k\frac{\partial T}{\partial x}\right)_{fi} = \frac{1}{\Delta x}\left[\left(k_f\frac{T_{f,i+m} - T_{f,i}}{x_{i+m} - x_i}\right) - \left(k_f\frac{T_{f,i} - T_{f,i-m}}{x_i - x_{i-m}}\right)\right] \tag{38}$$

which results in:

$$\left(\frac{\partial}{\partial x}k\frac{\partial T}{\partial x}\right)_{fi} = \frac{1}{\Delta x}\left[\left(\tilde{k}_{f,i+1/2}\frac{T_{f,i+m} - T_{f,i}}{x_{i+1} - x_i}\right) - \left(\tilde{k}_{f,i-1/2}\frac{T_{f,i} - T_{f,i-m}}{x_i - x_{i-1}}\right)\right] \tag{39}$$

Where effective thermal conductivity is defined as,

$$\begin{aligned}\tilde{k}_{f,i\pm1/2} &= \frac{k_f}{\max\left(\varepsilon, \frac{x_i - x_{i\pm m}}{x_i - x_{i\pm1}}\right)} \quad \text{if} \quad \Phi_i \Phi_{i\pm1} \le 0 \\ &= k_f \qquad\qquad\qquad\qquad\quad \text{if} \quad \Phi_i \Phi_{i\pm1} > 0\end{aligned} \tag{40}$$

Close to the interface , the temperature field needs to be extrapolated in order to define ghost points. Possible ways to extrapolate using constant, linear and quadratic extrapolation are (Gibou *et al.*, 2001):

$$\begin{aligned}T_{i+1}^G &= T_I \\ T_{i+1}^G &= \frac{T_I + (\theta - 1)T_i}{\theta} \\ T_{i+1}^G &= \frac{2T_I + (2\theta^2 - 2)T_i + (-\theta^2 + 1)T_{i-1}}{\theta^2 + \theta}\end{aligned} \tag{41}$$

where,

$$\theta = \frac{X_I - X_i}{\Delta x} = \frac{|\phi|}{\Delta x}$$

For the present study, the ghost temperature field is populated using constant extrapolation as it effectively moves the interface to the closest grid point while the remaining two extrapolation techniques behave poorly for small θ. As cited in (Gibou *et al.*, 2001) this second order accurate perturbation does not degrade the overall second order accuracy of the solution as long as the desired solution has bounded first derivatives. Temperature gradient is discretized as (Figure 5):

$$\left(\frac{\partial T}{\partial x}\right)_{fi} = \left(\frac{\partial T}{\partial x}\right)_{fi}^{-} \quad \text{if } \Phi_i\Phi_{i-1} \le 0 \text{ or } \left|\Phi_{i-1}\right| < \left|\Phi_{i+1}\right|$$

$$= \left(\frac{\partial T}{\partial x}\right)_{fi}^{+} \quad \text{if } \Phi_i\Phi_{i+1} \le 0 \text{ or } \left|\Phi_{i-1}\right| > \left|\Phi_{i+1}\right|$$

$$= \frac{1}{2}\left[\left(\frac{\partial T}{\partial x}\right)_{fi}^{+} + \left(\frac{\partial T}{\partial x}\right)_{fi}^{+}\right] \quad \text{if } \Phi_i\Phi_{i-1} \le 0 \text{ and } \Phi_i\Phi_{i-1} \le 0$$

Where,

$$\left(\frac{\partial T}{\partial x}\right)_{fi}^{\pm} = \frac{T_{fi} - T_{f,i\pm m}}{x_i - x_{i\pm m}} \quad \text{if} \quad \frac{x_i - x_{i\pm m}}{x_i - x_{i\pm i}} \ge 0.01$$

$$= \frac{T_{f,i\mp m} - T_{f,i\pm m}}{x_{i\mp m} - x_{i\pm m}} \quad \text{otherwise} \tag{42}$$

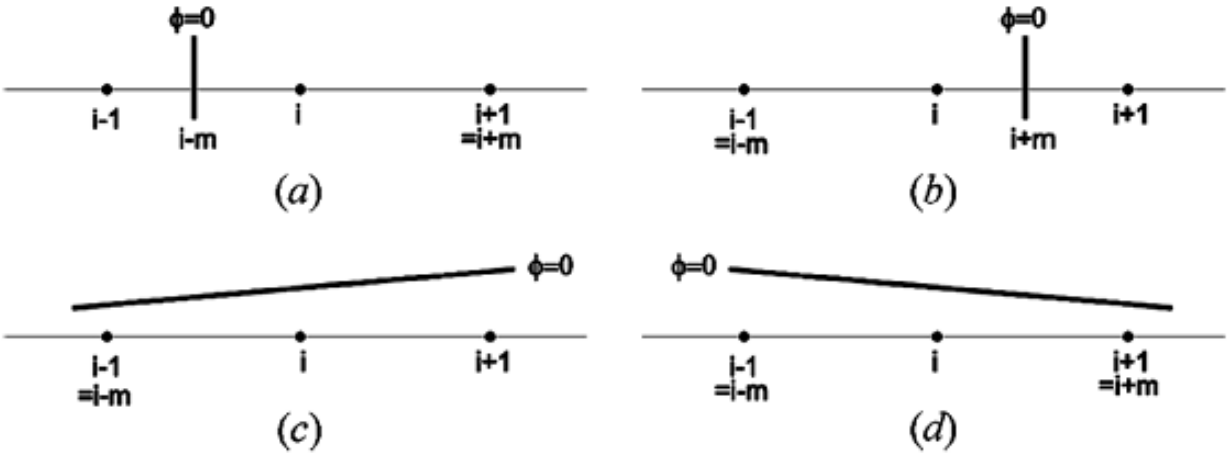


Figure 5 Discretization near the interface [Son and Dhir (2008)]

The computed $\partial T_f/\partial n$ is then extrapolated along the characteristics that flow outward from the interface by solving the following Hamilton-Jacobi equation.

$$\frac{\partial}{\partial \tau}\left(\frac{\partial T_f}{\partial n}\right) + sign(\Phi)\vec{n}.\nabla\left(\frac{\partial T_f}{\partial n}\right) = 0 \tag{43}$$

Rather than converging the above equation to a steady state, an alternative static Hamilton-Jacobi equation of the following form is solved:

$$\nabla\left(\frac{\partial T_f}{\partial n}\right).\nabla\Phi = 0 \tag{44}$$

The above equation can also be solved using central differencing scheme but upward differencing is employed as information is propagated along the characteristics (Osher and Fedkiw, 2002). The fast marching method (FMM) yields the same result as PDE based extrapolation and can be extended to higher order extrapolation analogous to the PDE based approach. (Aslam 2003, McCaslin *et al.* 2014). Mass flux is then evaluated from the energy balance at the interface by:

$$\dot{m} = \frac{1}{h_{lv}}\vec{n}.\left(k_l\nabla T_l - k_v\nabla T_v\right) \tag{45}$$

The mass flux evaluated at the interface is then further populated across a narrow band on both sides of the interface to be used in the computation of ghost fluid cells. Level set equations are discretized by using a second-order, essentially nonoscillatory (ENO) scheme described as:

$$U\left(\frac{\partial \Phi}{\partial x}\right)_i = \max\left(U_i, 0\right)\left(\frac{\partial \Phi}{\partial x}\right)_i^- + \min\left(U_i, 0\right)\left(\frac{\partial \Phi}{\partial x}\right)_i^+$$

$$\left(\frac{\partial \Phi}{\partial x}\right)_i^2 = \max\left[s\left(\frac{\partial \Phi}{\partial x}\right)_i^-, -s\left(\frac{\partial \Phi}{\partial x}\right)_i^+, 0\right]^2$$

$$\left(\frac{\partial \Phi}{\partial x}\right)_i^- = \left(\frac{\Phi_i - \Phi_{i-1}}{h}\right) + 0.5h \min \bmod \left(\frac{\Phi_{i+1} + \Phi_{i-1} - 2\Phi_i}{h^2}, \frac{\Phi_i + \Phi_{i-2} - 2\Phi_{i-1}}{h^2}\right)$$

$$\left(\frac{\partial \Phi}{\partial x}\right)_i^+ = \left(\frac{\Phi_i - \Phi_{i-1}}{h}\right) - 0.5h \min \bmod \left(\frac{\Phi_{i+1} + \Phi_{i-1} - 2\Phi_i}{h^2}, \frac{\Phi_i + \Phi_{i+2} - 2\Phi_{i+1}}{h^2}\right) \quad (46)$$

Where,

$$U_i = \frac{U_{i+1/2} + U_{i-1/2}}{2}, s = sign(S)$$

$$\min \bmod(a,b) = sign(a) \min\left(|a|, |b|\right) \quad if \ ab > 0$$

$$= 0 \qquad otherwise$$

Adaptive time stepping is used where the overall time step is the minimum of incompressible time step and level set time step. A CFL restriction of 0.5 is chosen.

$$\Delta t = \min\left\{\Delta t^L, \frac{0.5}{\max(\frac{|u_{\max}|}{\Delta x}, \frac{|v_{\max}|}{\Delta y}, \frac{|w_{\max}|}{\Delta z})}\right\} \quad (47)$$

2.6 Method of Solution

The set of linear algebraic equations obtained leads to a system with a tridiagonal matrix of coefficients. Direct methods of solving the discretization equations for 2 dimensions (or 3 dimensions) require excessive storage and computation time. Since the set of nominally linear equations must be solved after every iteration the computation cost of direct solution is

unacceptable. Conjugate gradient method being the most prominent iterative method for solving sparse system of linear equations is used. The speed of convergence for the pressure Poisson equation is further enhanced by implementing the block correction procedure (Patankar (1981); Settari and Aziz (1973)). The coefficient matrix A is known and is symmetric positive definite matrix. With knowing the coefficient matrix A, residual vector b, and the scalar Φ, preconditioner M, maximum number of iteration i_{max}, error tolerance ϵ, the linear algebraic equations were solved using preconditioned conjugate gradient algorithm is given in next page. For computation of velocities and temperature field Gauss Siedel, Jacobi or TDMA is used as a preconditioner while Multigrid is used as a preconditioner for Pressure computations. The PCG algorithm is defined in Figure 6:

$i \Leftarrow 0$

$r \Leftarrow b - A\Phi$

$d \Leftarrow M^{-1} r$

$\delta_{new} \Leftarrow r^T d$

$\delta_0 \Leftarrow \delta_{new}$

while $i < i_{\max}$ and $\delta_{new} > \varepsilon^2 \delta_0$

{

$q \Leftarrow Ad$

$\alpha \Leftarrow \dfrac{\delta_{new}}{d^T q}$

$\Phi \Leftarrow \Phi + \alpha d$

If i is divisible by 50 then

r $\Leftarrow b - A\Phi$

else

$r \Leftarrow r - \alpha q$

$s \Leftarrow M^{-1} r$

$\delta_{old} \Leftarrow \delta_{new}$

$\delta_{new} \Leftarrow r^T s$

$\beta \Leftarrow \dfrac{\delta_{new}}{\delta_{old}}$

$d \Leftarrow s + \beta d$

$i \Leftarrow i + 1$

}

Figure 6 PCG Algorithm (Shewchuk, 1994)

2.7 Microlayer

The microscopic region where liquid-vapor interface meets the wall is of tremendous importance in boiling heat transfer also known as microlayer. Its existence under a bubble was first confirmed experimentally by Cooper and Llyod (1969). In this region the adhesion forces between the fluid molecules and the wall molecules are important as the liquid film consists of only few molecular layers thick (of the order of few nanometers) and doesn`t evaporate due to adhesion forces. This region is called the adsorbed film. As the radial distance is increased the effect of the adhesion forces is decayed and the macroscopic hydrodynamics governs the interface shape. As the thickness of the liquid film grows the value of the interface curvature gets high leading to high pressure jump across the interface. The local saturation temperature is different than the corresponding pressure in the macroscopic region thus changing the local thermodynamic equilibrium.

Figure **7** shows the microlayer formed under the bubble. Comprehensive analysis of microlayer including disjoining pressure, vapor recoil pressure and interfacial heat transfer resistance has been carried out by Lay and Dhir (1995).

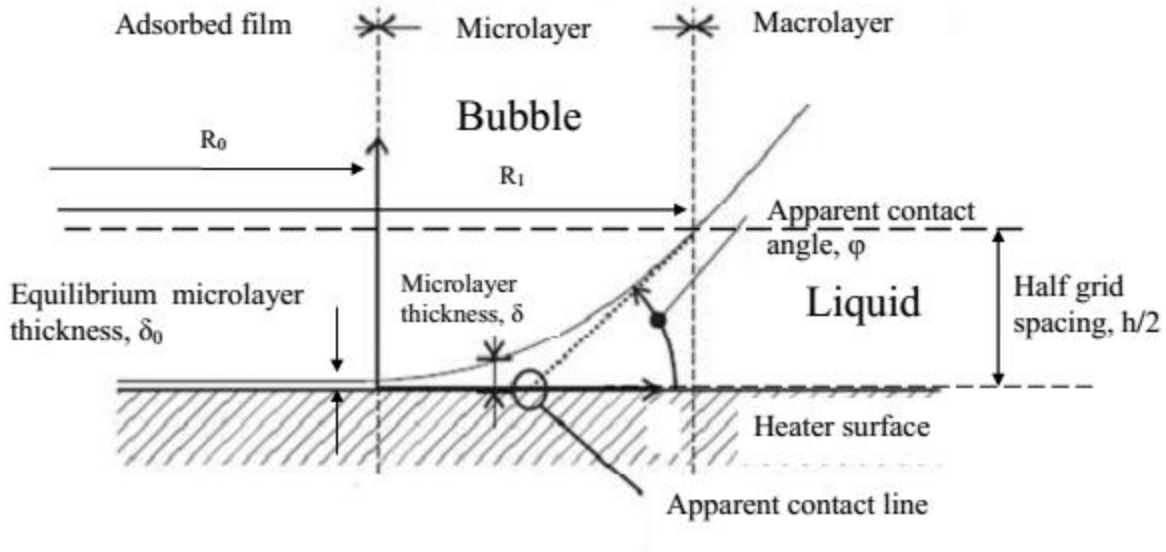


Figure 7 Microlayer under the bubble

According to the lubrication theory the conservation equation for mass, momentum and energy are given as:

$$\frac{\partial \delta}{\partial t} = v_l - \frac{q}{\rho_l h_{fg}} \tag{48}$$

$$\frac{\partial p_l}{\partial r} = \mu_l \frac{\partial^2 u_l}{\partial y^2} \tag{49}$$

$$q = \frac{k_l \left(T_w - T_{\text{int}}\right)}{\delta} = h_{ev}\left[T_{\text{int}} - T_v + \frac{(p_l - p_v)T_v}{\rho_l h_{fg}}\right] \tag{50}$$

The pressure in the vapor and liquid phase are related as

$$P_l = P_v - \sigma k - \frac{A}{\delta^3} + \frac{q^2}{\rho_v h_{lv}^2} \tag{51}$$

The combination of mass, momentum and energy equation gives a fourth order differential equation of the following form:

$$\delta'''' = f\left(\delta, \delta', \delta'', \delta'''\right) \tag{52}$$

The above fourth order differential equation can be cast into four first ordinary differential equations written as following:

$$\begin{aligned}
\frac{d\delta}{dr} &= \delta' \\
\frac{d\delta'}{dr} &= \frac{\left(1+\delta'^2\right)^{3/2}}{\sigma}\left(p_c - \frac{A}{\delta^3}\right) \\
\frac{dp_c}{dr} &= -\frac{3\mu_l Q}{\rho_l h_{fg} \delta^3} \\
\frac{dQ}{dr} &= \frac{T_{wall} - T_{sat}\left(1 + \dfrac{p_c}{\rho_l h_{fg}}\right)}{\dfrac{\delta}{k_l} + \dfrac{T_{sat}\sqrt{2\pi R_g T_{sat}}\left(2-\iota\right)}{h_{fg}^2 \rho_v 2\iota}}
\end{aligned} \tag{53}$$

The above ODE is solved in the direction from R_0 to R_1 with the following boundary conditions

$$\text{at } r = R_0 \qquad \delta = \delta_o, \delta' = 0, \delta''' = 0 \tag{54}$$

$$\text{at } r = R_1 \qquad \delta'' = 0 \tag{55}$$

δ_o is the equilibrium microlayer thickness, $\delta' = 0$ because the slope of the film is zero and $\delta''' = 0$ because of zero mass flow rate. R_0 and R_1 are related further by the following relation:

$$\tan\varphi = \left(\frac{h/2}{R_1 - R_0} \right) \tag{56}$$

With Hamaker;s constant, A, being unknown its values is iterated unless the height of the microlayer at R_1 is same as half of the grid spacing defined by the sixth condition defined as:

$$\delta = h/2 \tag{57}$$

The solution procedure of microlayer is as follows:

1. Obtain the value of R_1 from the macrolayer solution.
2. Calculate the value of R_0 from Equation 56.
3. Guess a value for Hamaker`s constant, A, and solve the fourth order ODE (Equation 53).
4. If Equation 57 is satisfied then microlayer solution is accepted else repeat step 3.

Figure 12 and Figure 13 shows the interface profile and heat flux near the contact line obtained from solution of the above mentioned fourth order ODE. It can be seen that local heat fluxes as high as 1.6 x 10^8 [W/cm^2] can exist near the contact line as previously reported by Lay and Dhir (1995). Two-dimensional quasistatic model of microlayer coupling with macrolayer which was first proposed by Son *et al.* (1999) is used as shown earlier in Figure 3 .The microlayer solution, $\dot{V}_{micro}$, obtained is evaluated by following equation as given by Son and Dhir (2008)

$$\dot{V}_{micro} = \int_{R_o}^{R_1} \frac{k_l \left(T_w - T_{sat}\right) dr}{\rho_v h_{lv} \delta \Delta V_{micro}} \tag{58}$$

3. Parallelization Framework

3.1 Introduction

Parallel computing has spearheaded research in computational sciences in an unparalleled way. A new era of multi-core computer has emerged which can be utilized by developing parallel applications. Requirement of sheer computational horsepower are increasingly met with tightly knitted integrated cluster systems comprising thousands of processors, petabytes of storage with high bandwidth/low latency of interconnects consuming megawatts of power, all of which are the paraphernalia of a parallel programming model thus abandoning the tradition of sequential algorithms.

For the comprehensive understanding of the multiphase phenomenon, the resolution of fluid-fluid interactions on very small scales is essential. This requires the usage of parallel computers to study the problem more accurately. The problem is solved on structured grid and domain decomposition method has been implemented using a distributed memory paradigm. Standard MPI protocols are used to communicate between the processors. Currently, the main domain is decomposed into three or more sub-domains in z direction in total. The parallel code is executed on UCLA`s Hoffman2 cluster which has 12,516 cores with Ethernet and infiniband interconnects.

Numerical solution of PDEs almost always implies a numerical lattice as the description of domain from which whole solution is sought. The two-dimensional domain has been decomposed into horizontal domains spanning along the vertical direction while the three dimensional domain

is decomposed into vertically sliced sub-domains. Each block (or sub-domain) is assigned to one processor (Figure 8). Each processor computes for its block and exchanges halo points (ghost points) with the neighboring processors when required. The value of halo points are updated between the iterations. Since, the width of the finite difference stencils employed in the present implementation ranges up to two grid cells, therefore, upto two slices of boundary ghost cells have to be attached at each domain. The boundary ghost values must be sent by the neighboring processors during the communication phase in the time stepping loop. Section 3.2 to section 3.4 explains the parallelization of the two-dimensional code while section 3.5 explains the parallelization of the three dimensional version.

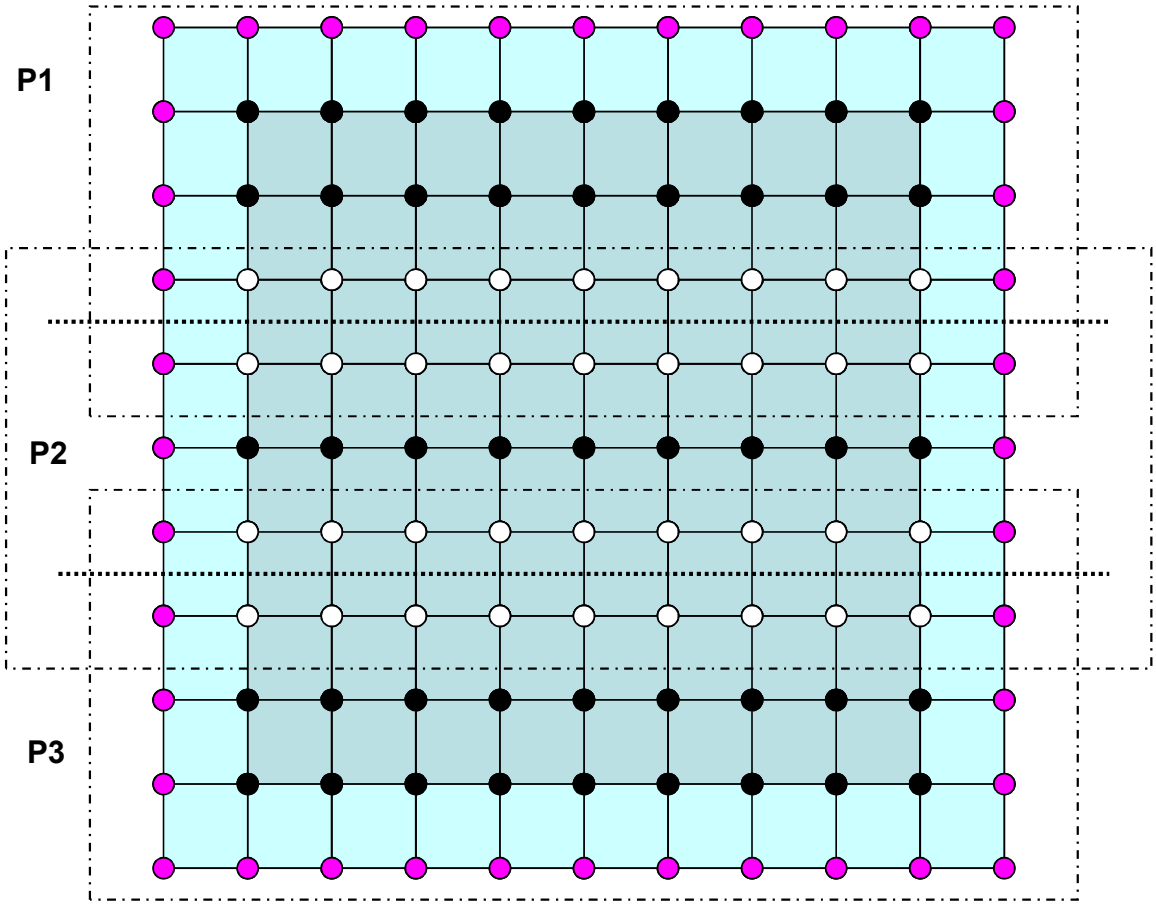


Figure 8 Parallelization by domain decomposition

3.2 Parallelization of Tri-Diagonal Matrix Algorithm (TDMA)

Presently, there are several approaches for parallelization of the Tri-Diagonal Matrix Algorithm (TDMA). Some of these are listed as:

1. Partition Method
2. PPT Algorithm
3. PDD Algorithm
4. LU Pipelining Algorithm
5. PTH Method and PPD Algorithm

The algorithm developed by Mattor *et al.* (1995) is implemented and described in detail which is analogous to the inhomogeneous linear differential equation and can be generalized to 5-diagonal and higher banded systems. It is conceptually different from parallel factorization in the sense that no manipulations are performed on the original matrix before subdividing it into different processors. The method adopted is based on LU decomposition, which is more efficient than cyclic reduction as multiple systems of equations needs to be solved.

The problem involving N x N linear system is considered on a parallel computer with P processors.

$$\Lambda X = R$$

where,

$$\Lambda = \begin{bmatrix} B_1 & C_1 & & & & \\ A_2 & B_2 & C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & & C_{N-1} & \\ & & & A_N & B_N & \end{bmatrix} \tag{59}$$

$$X = \left[X_1 \;\; X_2 \;\; X_3 \ldots\ldots\ldots X_N \right]^T$$

$$R = \left[R_1 \;\; R_2 \;\; R_3 \ldots\ldots\ldots R_N \right]^T \tag{60}$$

The linear system of order N is subdivided into P subsystems of order M (N=PM, for simplicity). Thus, the N x N matrix Λ is divided into P submatrices L_p, each of dimension M x M.

$$\Lambda = \begin{bmatrix} L_1 & c_M^1 e_M e_1^T & & & \\ a_M^1 e_1 e_M^T & L_2 & c_M^2 e_M e_1^T & & \\ & \ddots & \ddots & \ddots & \\ & & & & c_M^{P-1} e_M e_1^T \\ & & & a_1^P e_1 e_M^T & L_P \end{bmatrix} \tag{61}$$

Where, e_p is the p-th column of the M x M identity matrix. Similarly, vectors X and R are subdivided into P components as:

$$X = \left[X_1 \;\; X_2 \;\; X_3 \ldots\ldots\ldots X_P \right]^T$$

$$R = \left[R_1 \;\; R_2 \;\; R_3 \ldots\ldots\ldots R_P \right]^T \tag{62}$$

The sub-matrix L_p is given as:

$$L_p = \begin{bmatrix} B_{M(p-1)+1} & C_{M(p-1)+1} & & & \\ A_{M(p-1)+2} & B_{M(p-1)+2} & C_{M(p-1)+2} & & \\ & \ddots & \ddots & \ddots & \\ & & & C_{Mp-1} & \\ & & & A_{Mp} & B_{Mp} \end{bmatrix} = \begin{bmatrix} b_1 & c_1^p & & & \\ a_2^p & b_2^p & c_2^p & & \\ & \ddots & \ddots & \ddots & \\ & & & & c_{M-1}^p \\ & & & a_M^p & b_M^p \end{bmatrix}$$

(63)

For each subsystem p, three vectors X_p^R, X_p^{UH} and X_p^{LH} are obtained by solving the following system of equations:

$$\begin{aligned} L_p X_p^R &\equiv r_P \\ L_p X_p^{UH} &\equiv \begin{bmatrix} -a_1^p & 0 & 0 & \ldots\ldots & 0 \end{bmatrix}^T \\ L_p X_p^{LH} &\equiv \begin{bmatrix} 0 & 0 & \ldots\ldots & 0 & -c_M^P \end{bmatrix}^T \end{aligned} \tag{64}$$

The subscripts on the X stand for particular solution, upper homogeneous solution and lower homogeneous solution. The general solution of subsystem P is obtained by the following combination:

$$X_p = X_p^R + \xi_p^{UH} X_P^{UH} + \xi_P^{LH} X_P^{LH} \tag{65}$$

Where ξ_p^{UH} and ξ_p^{LH} are undetermined coefficients that depend on coupling to the neighboring solutions, and are obtained by solving the following system of equations:

$$\begin{bmatrix} X_{1,M}^{LH} & -1 & & & & & & \\ -1 & X_{2,1}^{UH} & X_{2,1}^{LH} & & & & & \\ & X_{2,M}^{UH} & X_{2,M}^{LH} & -1 & & & & \\ & & -1 & X_{3,1}^{UH} & X_{3,1}^{LH} & & & \\ & & & X_{3,M}^{UH} & X_{3,M}^{LH} & -1 & & \\ & & & & & X_{P-1,M}^{UH} & -1 & \\ & & & & -1 & & X_{P,M}^{UH} & \end{bmatrix} \begin{bmatrix} \xi_1^{LH} \\ \xi_2^{UH} \\ \xi_2^{LH} \\ \xi_3^{UH} \\ \xi_3^{LH} \\ . \\ . \\ \xi_{P-1}^{LH} \\ \xi_P^{UH} \end{bmatrix} = - \begin{bmatrix} X_{1,M}^{R} \\ X_{2,1}^{R} \\ X_{2,M}^{R} \\ X_{3,1}^{R} \\ X_{3,M}^{R} \\ . \\ . \\ X_{P-1,M}^{R} \\ X_{P,M}^{R} \end{bmatrix} \tag{66}$$

3.2 Parallelization of Multigrid

Multigrid algorithms are fast and efficient means of solving a variety of integral and partial differential equations by accelerating the convergence especially when applied to elliptic and parabolic equations. It requires that the problem is solved on a hierarchy of grids with different mesh sizes. With ever increasing demand for high resolution PDEs the parallelization of Multigrid for getting asymptotic performance is critical. Parallel Multigrid yields efficient schemes as long as there is sufficient computation to be performed by each processor. Nevertheless, on many parallel architecture, communication latencies are high compared to processor speed making coarse grid calculation dominated by communication. There are mostly four broad categories under which Multigrid can be run in parallel (Chow *et al.* 2006). These are concurrent iterations, multiple coarse corrections, full domain partitioning and block factorizations. For partitioning, the fine grid is usually subdivided ignoring the coarse mesh divisions. For the present problem, in

order to reduce the inter-processor communication during prolongation and restriction operation, the coarse mesh is derived from the fine mesh partition . The LNTDMA smoother used at every level is parallelized at every level using the decomposition in X and Y direction alternatively as shown in Figure 10. For a general linear equation written in the following form (for the two-dimensional case):

$$a_{i,j}\Phi_{i-1,j} + b_{i,j}\Phi_{i,j} + c_{i,j}\Phi_{i+1,j} + e_{i,j+1}\Phi_{i,j+1} + f_{i,j-1}\Phi_{i,j-1} = d_{i,j} \tag{67}$$

For sweeping in the X direction, the above equation is modified in the following form:

$$a_{i,j}\Phi_{i-1,j} + b_{i,j}\Phi_{i,j} + c_{i,j}\Phi_{i+1,j} = d_{i,j} - e_{i,j}\Phi_{i,j-1} - f_{i,j}\Phi_{i,j+1} \tag{68}$$

Which can be further cast into two similar equation as:

$$\begin{aligned} a_{i,j}\Phi_{i-1,j} + b_{i,j}\Phi_{i,j} + c_{i,j}\Phi_{i+1,j} &= d'_{i,j} \\ e_{i,j}\Phi_{i,j-1} + b_{i,j}\Phi_{i,j} + c_{i,j}\Phi_{i,j+1} &= d''_{i,j} \end{aligned} \tag{69}$$

for i=*iist* until *iiend* do {Local 'i' loop for every processor}
for j = 2 step until NJ do

$$m = \frac{a_{i,j}}{b_{i,j-1}}$$

$$b_{i,j} = b_{i,j} - m c_{i,j}$$

$$d'_{i,j} = d'_{i,j} - m d'_{i,j-1}$$

end loop (j)
Backward substitution phase

$$\Phi_{i,j} = \frac{d'_{i,j}}{b_{i,j}}$$

for j = NJ-1 stepdown until 1 do

$$\Phi_{i,j} = \frac{d'_{i,j} - c_{i,j}\Phi i,_{j+1}}{b_{i,j}}$$

end loop (j)
end loop (i)

for j=*jjst* until jj*end* do {Local 'j' loop for every processor}
for i = 2 step until NI do

$$m = \frac{a_{i,j}}{e_{i,j-1}}$$

$$e_{i,j} = e_{i,j} - m f_{i,j}$$

$$d''_{i,j} = d''_{i,j} - m d''_{i,j-1}$$

end loop (i)
Backward substitution phase

$$\Phi_{i,j} = \frac{d''_{i,j}}{b_{i,j}}$$

for j = NJ-1 stepdown until 1 do

$$\Phi_{i,j} = \frac{d''_{i,j} - c_{i,j}\Phi i,_{j+1}}{b_{i,j}}$$

end loop (i)
end loop (i)

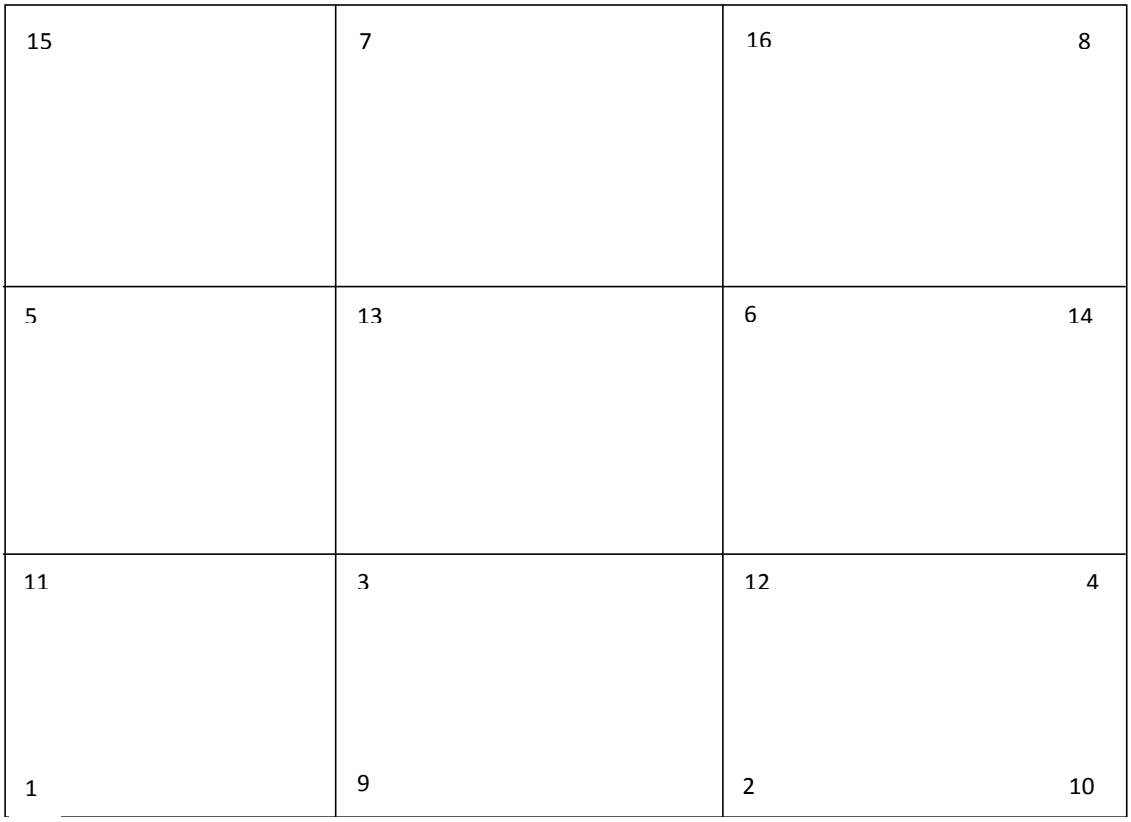


Figure 9 Red-black ordering of Gauss-Siedel algorithm

For parallelization of Gauss-Siedel algorithm, commonly used red-black ordering (Figure 9) is described next (Zhu, 1994). The coefficient matrix A in

$$A\Phi = B \tag{70}$$

has the following form,

$$A = \begin{bmatrix} D_b & E \\ E^T & D_r \end{bmatrix} \tag{71}$$

Where, D_b and D_r are diagonal matrices with -4 on the main diagonal while E is a sparse matrix with only 1s as its entries. Applying the Gauss-Siedel iteration to the unknown :

$$\begin{bmatrix} D_b & 0 \\ E^T & D_r \end{bmatrix} \begin{bmatrix} \Phi_b \\ \Phi_r \end{bmatrix}^k = \begin{bmatrix} 0 & E \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Phi_b \\ \Phi_r \end{bmatrix}^{k-1} + \begin{bmatrix} B_b \\ B_r \end{bmatrix} \tag{72}$$

The above equation can be separated as:

$$\begin{aligned} D_b \Phi_b^k &= E\Phi_r^{k-1} + B_b \\ E^T \Phi_r^k + D_r \Phi_r^k &= B_r \end{aligned} \tag{73}$$

Also, D_B is a diagonal matrix with -4 on the main diagonal, which yields:

$$\Phi_b^k = -\left[\frac{E\Phi_r^{k-1} + B_b}{4} \right] \tag{74}$$

The above equation means that all components of $\Phi_b{}^k$ can be calculated simultaneously following which all the red unknowns can be calculated as:

$$\Phi_r^k = -\left[\frac{B_r - E^T \Phi_b^k}{4}\right] \tag{75}$$

The completed Gauss Siedel relaxation scheme can be given as:

Initialize error = $1e^6$

While (error> tolerance)

1. Update all the black unknown points.

2. Communicate the black unknown points to the neighboring processors.

3. Update all the red unknown points.

4. Communicate the red unknown points to the neighboring processors.

5. Calculate error.

End

The SOR algorithm can also be parallelized in a similar way.

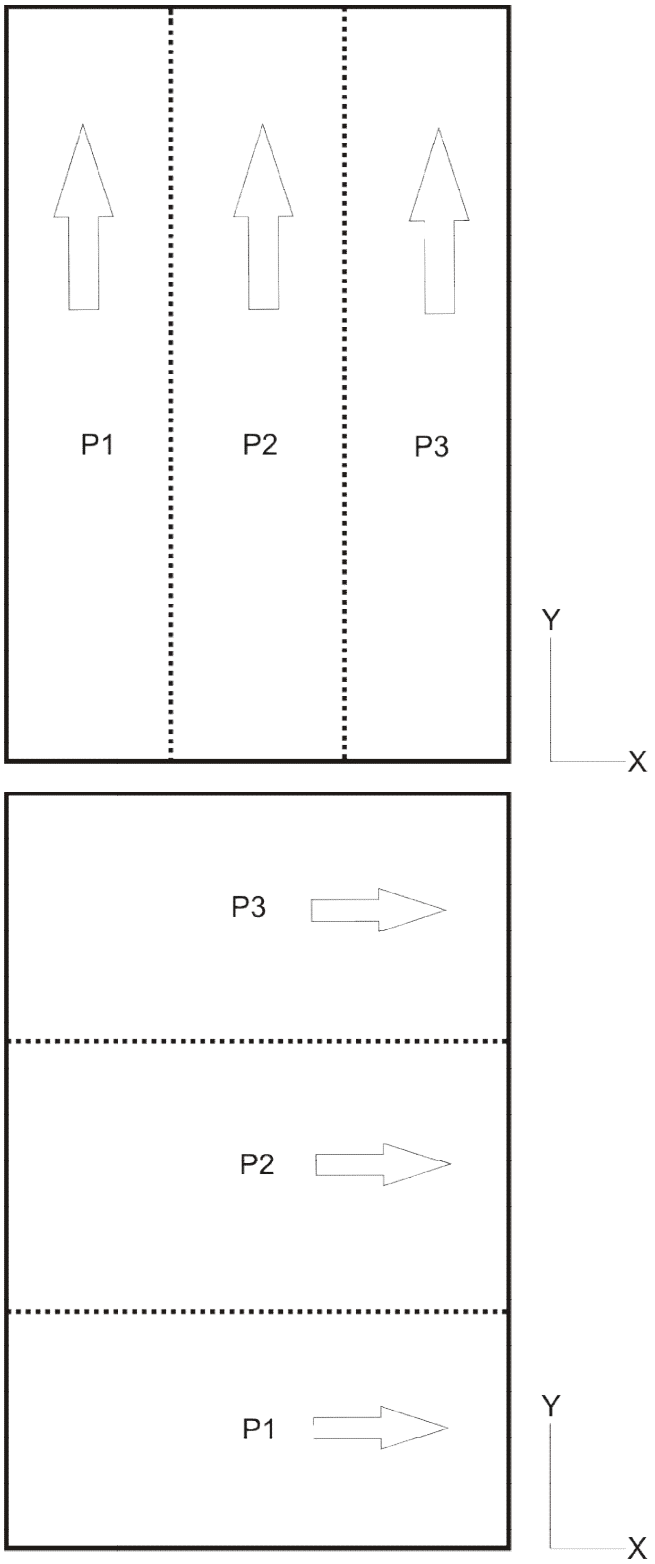


Figure 10 Domain decomposition in vertical and horizontal directions

3.4 Parallel Two Phase Solver

The parallel framework is summarized in the following algorithm:

Initialization

Initialize level set function with the bubble embryo.

Initialize temperature profile throughout the domain with initial thermal boundary layer.

Time Loop

While current time ≤ final time:

Time step restriction.

Compute local time step, δt.

Solve for Level set function (Φ).

Exchange neighbor values of level set function.

Compute ghost values of velocity (U,V,W).

Exchange neighbor values of velocity and temperature.

Solve energy equation to compute temperature (T).

Exchange neighbor values of temperature.

Compute mass flux across the interface.

Extrapolate mass flux at the interface to a narrow band using Fast Marching Method (FMM).

Solve momentum equation to compute velocities.

Exchange neighbor values of velocities.

Solve pressure Poisson equation with intermediate U,V,W to compute P.

Exchange neighbor values of P.

Projection of velocity to divergence free space.

Exchange neighbor values of U,V,W.

Call boundary conditions of U,V,W.

Increment $t = t + \delta t$.

3.5 Parallelization of the coefficient matrix for the 3D two phase solver

The parallelization approach described in section 3.2 to 3.4 was further supplemented using hypre library (for parallelization of the three-dimensional code only). Hypre is a software library for high performance preconditioners and solvers for the solution of large, sparse system of linear equations suitable for massively parallel computers. It supports four conceptual interfaces as follows:

1. Structured-grid system interface
2. Semi-structured grid system interface
3. Finite element interface
4. Linear-algebraic system interface

Out of the above four interfaces, the Linear-algebraic system interface (IJ interface) was used as it was possible to couple with the existing three-dimensional code to the IJ interface without changing rest of the code, to enable code scaling upto thousands of processors. The problem data is passed into this interface in the distributed form and matrices are assumed to be distributed in the following form:

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ . \\ . \\ . \\ A_{p-1} \end{bmatrix} \tag{76}$$

The resulting system of equations can be solved using variety of preconditioners, some of which are explained in the next section.

3.5.1 ParaSails

ParaSails is a parallel implementation of a sparse approximate inverse preconditioner, using *a priori* sparsity patterns and Frobenius norm least-squares minimization. Additionally it uses a post-filtering technique to reduce the cost of applying the preconditioner. Symmetric positive definite (SPD) problems are handled using a factored SPD sparse approximate inverse while non-symmetric problems are handled with an unfactored sparse approximate inverse. More details about the ParaSails preconditioner can be found from Chow, E. (2000).

3.5.2 BoomerAMG

BoomerAMG is a parallel implementation of the algebraic multigrid method which can be used both as a solver as well as a preconditioner. Different parallel coarsening techniques, interpolation and relaxation schemes can be used. Some of the coarsening techniques that can be used are:

1. Cleary-Luby-Jones-Plassman (CJLP) coarsening.
2. Falgout coarsening, which is a combination of CJLP and RS coarsening algorithm.
3. CGC and CGC-E coarsening.
4. PMIS and HMIS coarsening.

Similarly, various interpolation techniques like classical, direct, standard, multipass, two stage and Jacobi, etc can also be used.

3.6 Conclusion

The parallelization techniques discussed in the present chapter aren`t sufficient to enable the entire code to be executed in parallel and hence speed up gained was limited to approximately 20% . TDMA, GS and Hypre Library were used interchangeably to solve the linear system of equations as no single technique was sufficient for different conditions like grid stretching, coarse grids etc . In future, rest of the code can be undertaken in parallel with implementation of advanced data structures and algorithms.

4. Two Dimensional Boiling Curve Simulation

4.1 Introduction

In carrying out two-dimensional numerical simulations, the properties of saturated water at 1 atm were used. The contact angles considered are 27º, 38º, 54º, 69º and 90º. In actuality, the contact angle varies dynamically between an advancing contact angle and a receding contact angle however, the effect of dynamic contact angle is not considered in the present work. The dimensionless computational domain for the present study is thus chosen to be $0 \leq x \leq 5$ and $0 \leq y \leq 50$ resolved by grid density of 386 x 386. The width of the computation domain has to be atleast 2.5 cm for simulating the entire boiling curve as the Taylor most dangerous wavelength (λ_D) corresponding to earth normal gravity is approximately 2.5 cm. The height of the domain is chosen to be large enough so that the bubble motion would not get affected by the top boundary of the computational domain. A non uniform grid spacing in the y direction is taken with the node expansion ratio of 1.01, except near the wall $0 \leq y \leq 0.45$, where the node spacing is uniform as Δy. The node spacing in the x direction is uniform throughout with Δx while the side boundaries at x = 0 and x = 5 are treated as symmetric. The non-dimensional governing equations of mass, momentum and energy are solved for both the liquid and vapor phases. The non-dimensionalized length (l_o), time (t_o) and velocity (u_o) scales are defined in terms of surface tension (σ), earth normal gravity (g), water density (ρ_l) and vapor density (ρ_v) as:

$$l_o = \sqrt{\frac{\sigma}{g\left(\rho_l - \rho_v\right)}},\ u_o = \sqrt{\left(g l_o\right)},\ t_o = \frac{l_o}{u_o} \tag{77}$$

The values of non-dimensionalized scales defined above are evaluated as l_o = 2.5 mm, t_o = 15.97 ms and u_o = 0.16 m/s. The liquid temperature profile is defined as linear in the natural convection thermal boundary layer and the fluid velocity is set equal to zero. The initial thermal

boundary layer thickness, δ_T ,is evaluated from the following turbulent natural convection heat transfer correlation:

$$\delta_T = 7.14\left(\frac{\mu_l \alpha_l}{\rho_l g \beta_T \Delta T}\right)^{1/3} \tag{78}$$

Since heat conduction in the solid phase is not computed, in order to simulate realistic surface bubble nucleation frequency and cavity density (*Na*) are obtained from correlations reported in the literature. There are many expressions for bubble waiting time which however require the knowledge of the cavity size and the thermal properties of the heater (Hsu and Graham, 1986, Kocamustafaogullari and Ishii, 1983). Therefore, the correlation proposed by Basu et al. (2005) (Equation 79, 80) is used for computing the waiting period and nucleation site density.

$$t_{wait} = \frac{139.1\Delta T^{-4.1}}{t_o} \tag{79}$$

$$Na\ (\text{sites/cm}^2) = 0.34\left[1-\cos\varphi\right]\Delta T^{2.0} \quad \text{for } \Delta T < 15\ \text{K}$$
$$Na\ (\text{sites/cm}^2) = 3.4\text{x}10^{-5}\left[1-\cos\varphi\right]\Delta T^{5.3} \text{for } \Delta T \geq 15\ \text{K} \tag{80}$$

To obtain the number of cavities for the two-dimensional case (of length L_x) from the actual experimental correlation, which is for three-dimensional case, the following relation is derived based on square grid arrangement of cavities in three-dimensional case:

$$N = L_X \sqrt{Na} \tag{81}$$

The value of L_x for the two-dimensional case is 1.25 cm. The properties of the water used for the present case are listed in Table 1.

Properties	**Value**	**Units**
Thermal conductivity of liquid	0.681	W/m-k
Thermal conductivity of vapor	0.0248	W/m-k
Surface tension	0.0589	N/m
Specific heat capacity of liquid	4212	J/kg-k
Specific heat capacity of vapor	2020	J/kg-k
Density of liquid	958	Kg/m^3
Density of vapor	0.5977	Kg/m^3
Kinematic Viscosity of liquid	3.0×10^{-7}	m^2/s
Kinematic Viscosity of vapor	2.01×10^{-5}	m^2/s
Thermal expansion coefficient	7.1×10^{-4}	1/k
Latent heat of vaporization	2.26×10^{6}	J/Kg

Table 1 Property values used for the current study

Contact Angle	ΔT_w [K]	Na (Basu) [Sites/cm^2]	Na (2D Case) [Sites/cm]	t_{wait}* [-]
27	10	4	2	0.69186636
27	15	6	3	0.13123451
27	20	29	7	0.04034588
27	24	76	11	0.0191054
27	28	173	16	0.01015487
27	30	250	20	0.00765288
38	10	7	3	0.69186636
38	15	12	4	0.13123451
38	20	57	9	0.04034588
38	24	149	15	0.0191054
38	28	337	23	0.01015487
38	30	485	28	0.00765288
54	10	14	5	0.69186636
54	15	24	6	0.13123451
54	20	110	13	0.04034588
54	24	289	21	0.0191054
54	28	655	32	0.01015487
54	30	944	38	0.00765288
69	10	22	6	0.69186636
69	15	37	8	0.13123451
69	20	171	16	0.04034588
69	24	450	27	0.0191054
69	28	1019	40	0.01015487
69	30	1469	48	0.00765288
90	10	34	7	0.69186636
90	15	58	10	0.13123451
90	20	267	20	0.04034588
90	24	702	33	0.0191054
90	28	1589	50	0.01015487
90	30	2290	60	0.00765288

Table 2 Waiting time and nucleation site density for different contact angles

Initially small bubble embryos are placed sequentially along the heater surface with nucleation time interval equal to the waiting time between cavities. The addition of new bubble depends on the waiting period thereafter. Cavities are evenly spaced along the length of the heater with the maximum limit of 100 cavities in order to keep the cavities separated by at least 3-4 grid points. Table 2 shows the number of cavities (rounded off) and the waiting time for different superheat considered for the current work till wall superheat of 30 °C. For the present work dry region at the heater surface during boiling process is considered by accounting the area of the adsorbed film under the bubble, corresponding to the radius R_0 from the cavity centre, as shown in Figure 11 whereas the wall heat flux and the heat flux going into the vapor phase are computed from the following two equations, where q_{mic} is the microlayer contribution to the wall heat flux computed separately near the contact line.

$$q_{wall} = \left(\frac{-k_{liq}\left(\frac{\partial T_f}{\partial y}\right)_{|y=0} A_{liq} \; - k_{vap}\left(\frac{\partial T_f}{\partial y}\right)_{|y=0} A_{vap} + \; q_{mic} A_{mic}}{A_{wall}} \right) \tag{82}$$

$$\vec{q}''_{vap} = \vec{n}.\left(k_l \nabla T_l - k_v \nabla T_v\right) \tag{83}$$

In Equation 82, A_{liq} is the liquid occupied area of the surface, A_{vap} is the dry area below the bubble excluding the microlayer area and A_{mic} is the microlayer area underneath the bubble calculated by multiplying the contact line length by the length of the microlayer. The microlayer profile and heat flux, with respect to distance from R_0, obtained by solving microlayer equations for contact angles 38° and 69° are shown in Figure 12 and Figure 13 respectively.

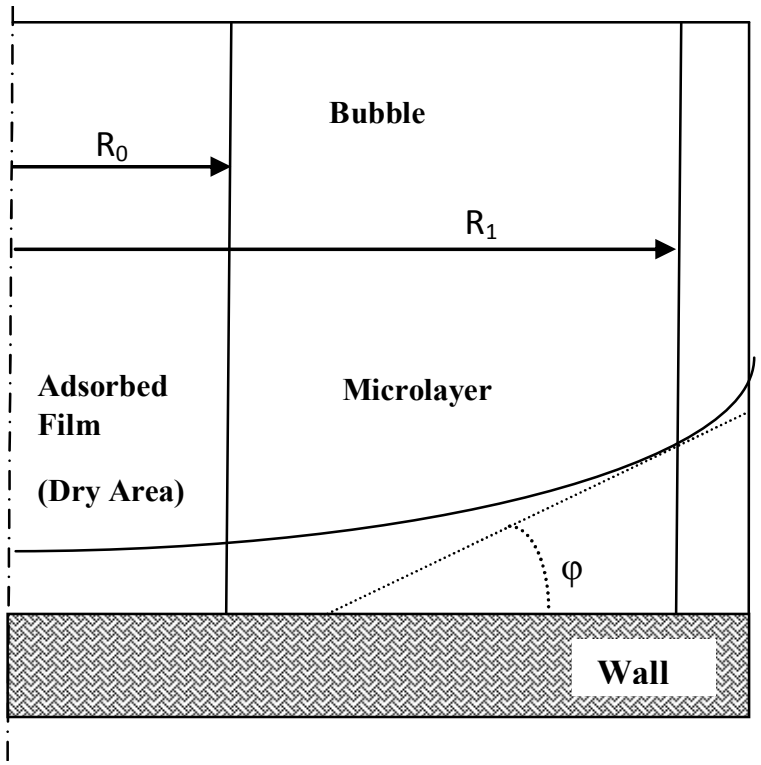


Figure 11 Dry area under the bubble.

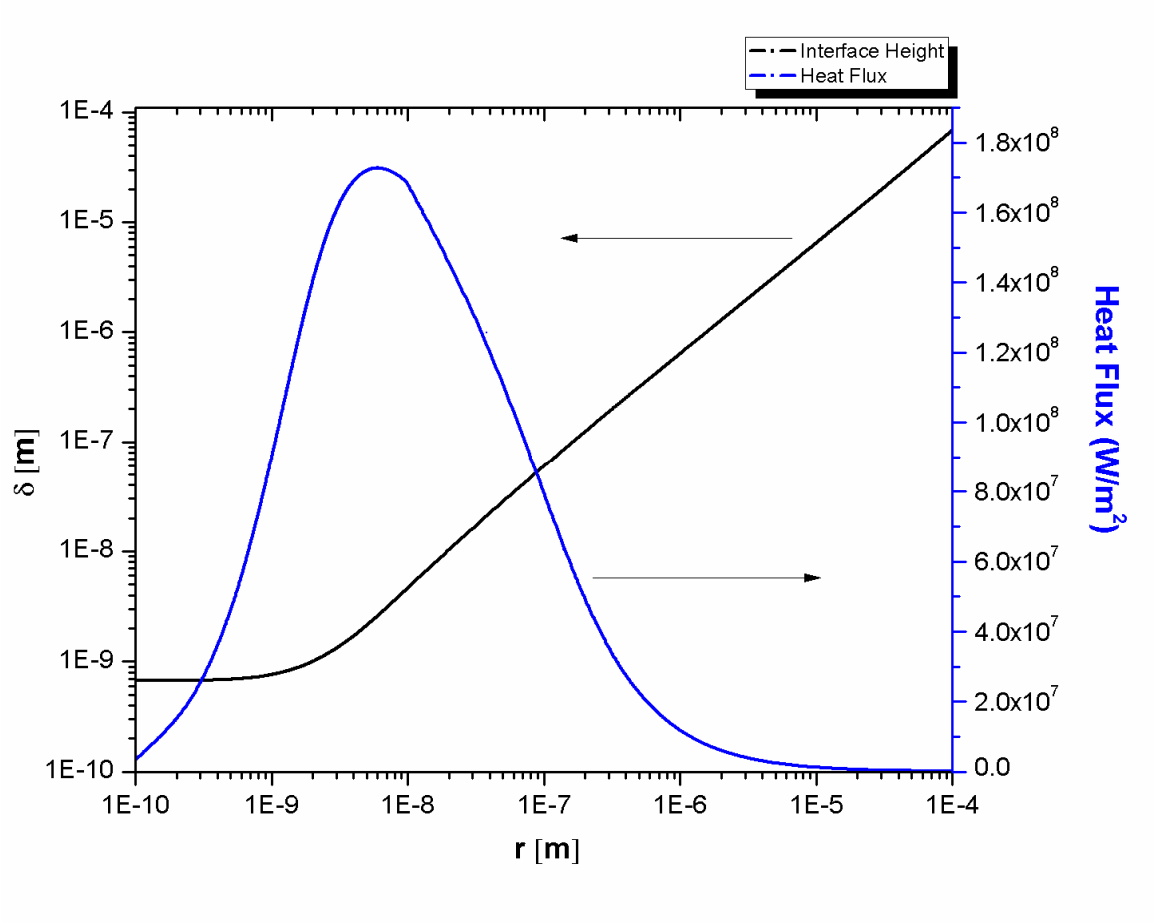


Figure 12 Interface profile and heat flux variation along microlayer ($\varphi = 38^\circ$ and $\Delta T = 12^\circ K$)

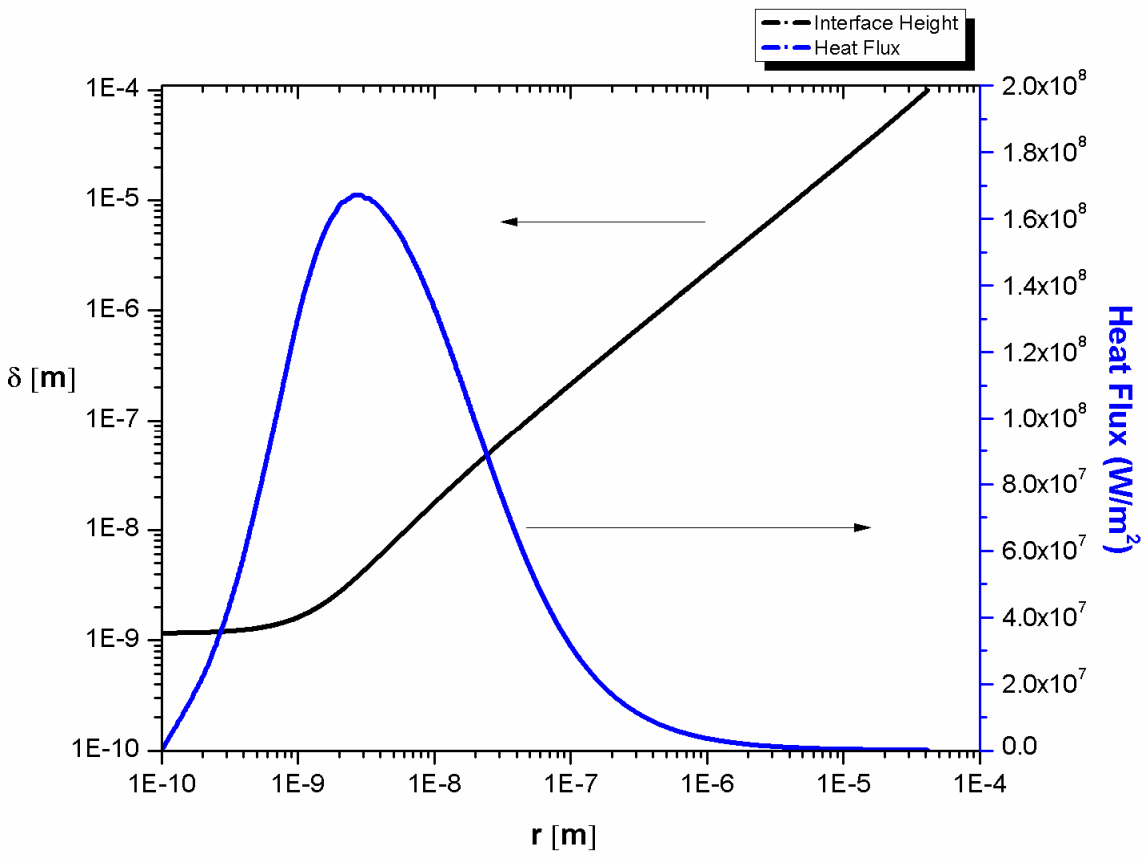


Figure 13 Interface profile and heat flux variation along microlayer ($\varphi = 69^{\circ}$ and $\Delta T = 12^{\circ}$ K)

4.2 Results and Discussion - Contact Angle 38°

Figure 14 - Figure 19 shows the bubble growth and merger pattern with liquid velocity vectors along with wall heat flux and wall void fraction variation for different superheats for the case of contact angle of 38º only Figure 14 shows the boiling process for wall superheat of 17 ºC (Point a in Figure 26). At t* = 0.30 all the cavities have nucleated with the bubbles still attached to the surface pushing the liquid out of the domain shown by upward pointing velocity vectors. As the simulation progresses, the periodical formation of liquid macrolayer formed due to liquid entrapment below lateral merger of the bubbles is observed during the whole period of vapor evolution. At t* = 6.00 bubbles have merged in both the vertical and horizontal directions creating an elongated vapor mass continuously being fed from bubbles nucleating from the surface. It can be seen at t* = 8.00 that the process has transitioned into a quasi-static state in which the same pattern is seen repeating multiple times as time proceeds. Figure 15 shows the temporal variation of wall void fraction and heat fluxes per unit surface area of the heater from the wall and into vapor production. When the heat fluxes are averaged over the entire area, except the early computation time ($t^* \geq 6$), it is found that about 50% of the energy from the wall is utilized in vapor production (including microlayer) while the other half is utilized in superheating the liquid. Since vapor is assumed to be saturated for the present case, energy utilized for superheating of vapor is zero.

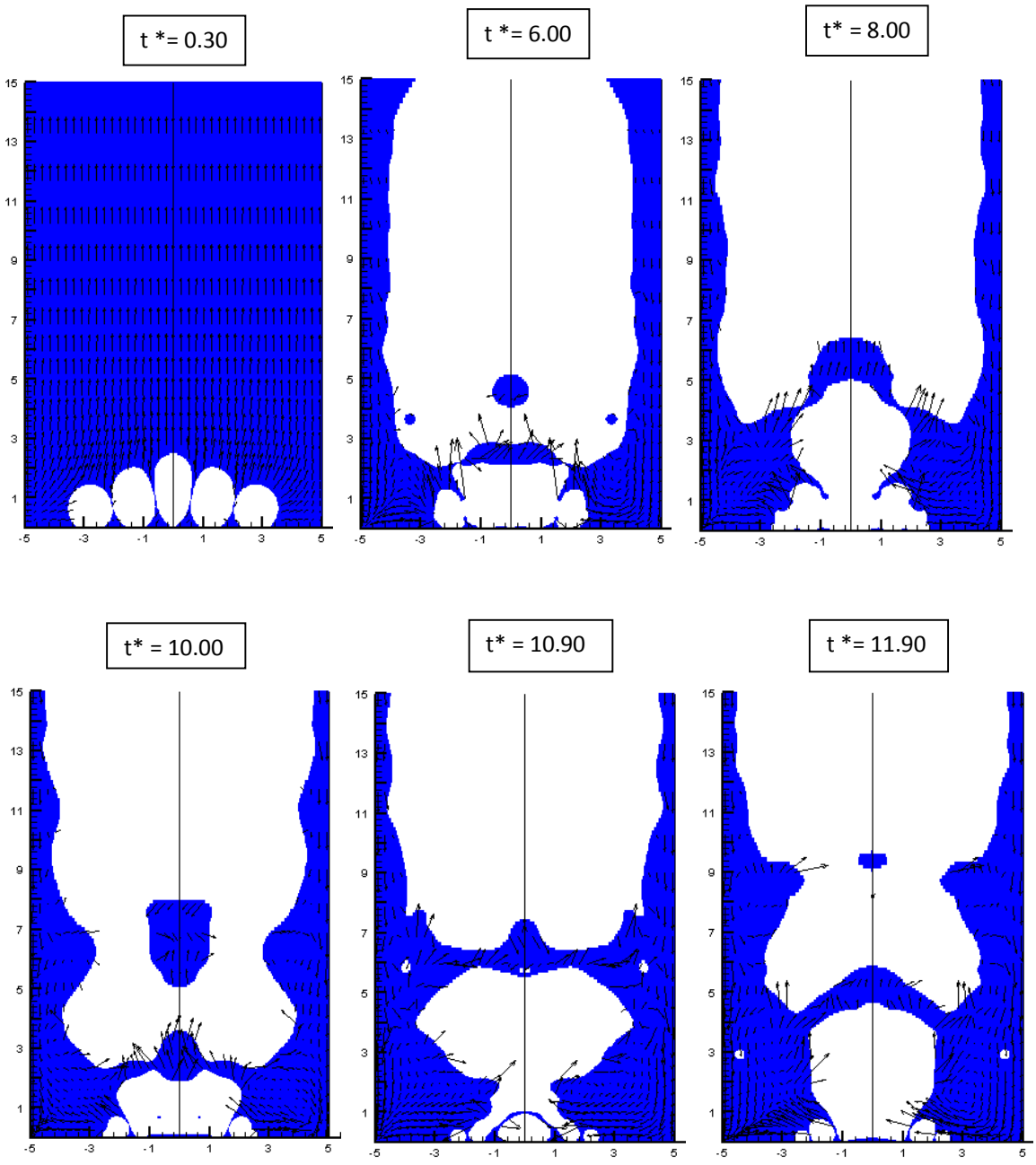


Figure 14 Boiling process at nucleate boiling (ΔT = 17 ºC)

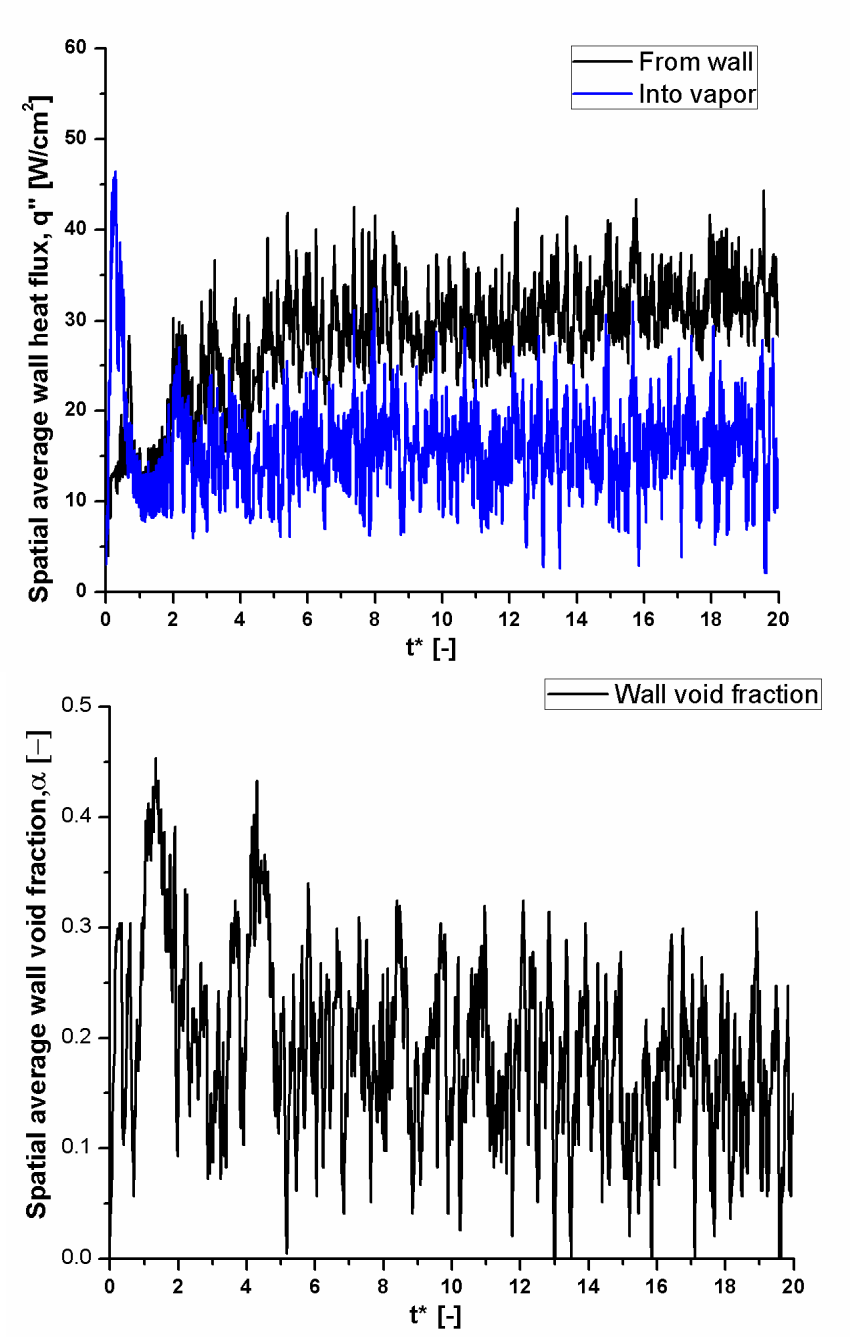


Figure 15 Temporal variation of heat fluxes and wall void fraction (Nucleate boiling, ΔT = 17 ºC)

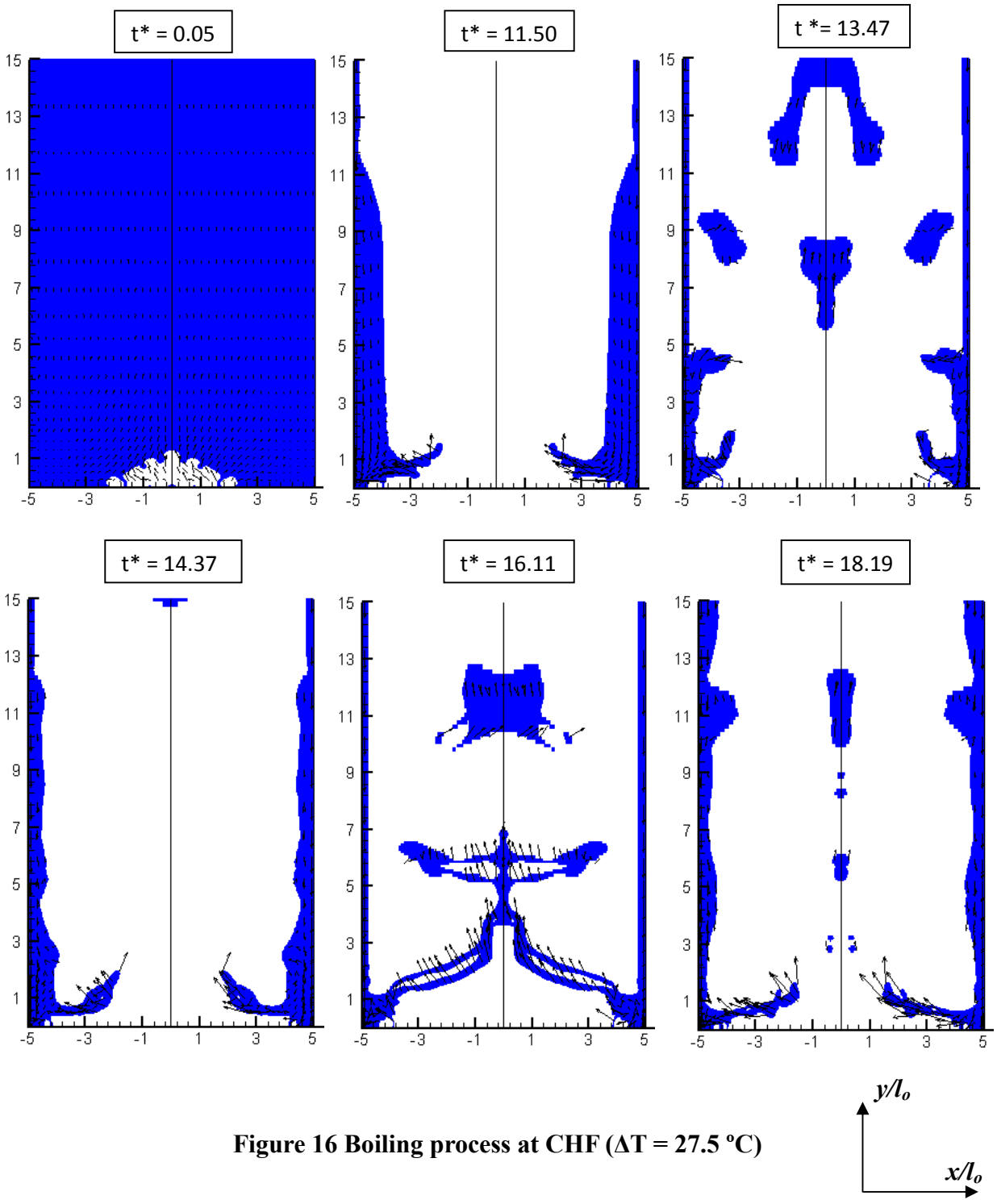


Figure 16 Boiling process at CHF (ΔT = 27.5 ºC)

Figure 16 shows the boiling process numerically simulated at CHF corresponding to wall superheat of 27.5 ºC (Point b in Figure 26). Long vapor columns or jets can be seen evolving at t* = 11.50 which constricts near to the surface. At t* = 13.47 multiple liquid droplets can be seen getting entrained into the vapor column which merge and break among themselves inside the vapor core. This pattern resembling at t* = 13.47 is seen repeating again at t* = 14.37, t* = 16.11 and at t* = 18.19. However, at t* = 16.11 liquid filament is seen inside the vapor column rather than droplets unlike t* = 14.37 and t* = 18.19. Figure 17 shows the temporal variation of wall void fraction and heat fluxes for this case. Except for the early simulation time, a drop in the wall void fraction at any time instant causes a spike in the wall heat flux. At t* = 4.00 a temporary drop in the wall void fraction to 0.56 causes the instantaneous spatial averaged wall heat flux to rise to 107.34 W/cm^2 showing negative correlation between wall heat flux and wall void fraction. Between t* = 4.00 to t* = 11.50 only minor oscillations in the heat flux is seen as the jet is still rising and wall void fraction is almost constant. After t* = 11.50 as the vapor exits the domain, liquid inflow occurs from the sides. Significant oscillations in the heat fluxes is seen thereafter as the jet breaks near the surface altering the wall void fraction. The temporal averaging is done after t* $\geq$ 12 over the entire area to compute wall heat flux and wall void fraction. Figure 18 shows the boiling process at transition regime corresponding to the wall superheat of 40 ºC (Point c in Figure 26). Initially much of the surface area is covered by vapor due to an increase in the active nucleation sites. The average vapor velocity is approximately 0.2 m/s. At t* = 6.67 instability of the vapor column can be seen close to the heater surface which causes it to detach and form an elongated bubble. At t* = 7.42 vapor slug is seen evolving from the surface and assimilating into the elongated bubble. This process can be seen repeating multiple times henceforth. Temporal variation of wall void fraction

and heat fluxes for the case of transition boiling is plotted in Figure 19. Due to the formation of stationary vapor stems trapping liquid macrolayer between them, no significant oscillation with time in the wall void fraction and thereby wall heat flux is observed (Figure 20). The average value of wall void fraction is computed to be 0.954 for the present case while the value of heat flux from the wall is 38.18 W/cm^2.

The understanding of the mechanism behind CHF is still unclear. Zuber`s (1958) hydrodynamic theory model, later extended by Lienhard and Dhir (1973), does not include the surface effects on maximum heat flux. However, there is substantial evidence that the maximum heat flux is influenced by the degree of wettability of the heater (Costello and Frea (1965), Hasegawa *et al.* (1973), Hahne and Diesselhorst (1978)). Table 3 shows the comparison of the values of critical heat flux, which for the current study is 78.9 W/cm^2 in comparison to the CHF value of 76.8 W/cm^2 obtained by Liaw and Dhir (1989), for the same contact angle but for vertical surface. CHF values of 80.3 W/cm^2 (φ = 37º) and 72.75 W/cm^2 (φ = 38º) were obtained experimentally for different surface material by Hahne and Diesselhorst (1978). Figure 27 shows the variation of wall void fraction with superheat, at CHF the value of wall void fraction computed is 0.72, implying that the liquid can access about 25% of the surface and the maximum heat flux, which is lower than the Zuber`s model, is due to the limitation on the energy removal mechanism on the liquid occupied region of the surface (Liaw and Dhir (1989)).

Auracher *et al.* (2004) performed experiments for well wetting fluids and fluids with larger contact angle under steady state and transient conditions, and did not observe any hysteresis in the transition region for a clean heater surface. For the present study hysteresis is not considered in the

transition boiling regime, which corresponds to region between wall superheat of 27.5 ºC and 110 ºC. In the current simulation the value of wall void fraction increases with increase in superheat implying that liquid contact with the surface has decreased. Theoretically using Zuber`s value of minimum heat flux (Equation 84), with the value of $C_2$ taken as 0.09, and Berenson`s correlation (Equation 85) for film boiling, the transition to film boiling occurs at a wall superheat of 85.2 ºC. However, for the present study since the computed value of wall heat flux at the wall superheat of 130 ºC is higher than that of 110 ºC it can be stated that 110 ºC is the wall superheat corresponding to minimum wall heat flux and wall heat flux starts increasing thereafter.

$$q''_{\min} = C_2 \rho_v h_{fg} \left[\frac{g\sigma(\rho_l - \rho_v)}{(\rho_l^2 + \rho_v^2)} \right]^{1/4} \tag{84}$$

$$Nu = 0.42 \left[\frac{\rho_v(\rho_l - \rho_v) g h_{fg}}{k_v \mu_v \Delta T} \right]^{1/4} \left[\frac{\sigma}{g(\rho_l - \rho_v)} \right]^{3/8} \tag{85}$$

For the case of film boiling, the value of heat flux is computed to be 0.42 W/cm^2 against 2.29 W/cm^2 given by Berenson`s correlation (for film boiling heat transfer coefficient) for a wall superheat of 110 ºC while a value of 1.21 W/cm^2 was obtained by Son and Dhir (1997) at a wall superheat of 100 ºC (Table 4).

Contact Angle	Surface	CHF [W/cm^2]	Reference
38 º	Horizontal	78.9 (Numerical)	Present work
38 º	Vertical	76.8 (Experimental)	Liaw and Dhir (1989)
35 º	Vertical	86.27(Experimental)	Wang and Dhir (1993)
37 º	Horizontal (CrNi-Steel)	80.3 (Experimental)	Hahne and Diesselhorst (1978)
38 º	Horizontal (Ta)	72.75 (Experimental)	Hahne and Diesselhorst (1978)
40 º	Horizontal	74.75 (Theoretical)	Kandlikar (2001)
45 º	Horizontal	76.5 (Experimental)	Maracy and Winterton (1988)

Table 3 Comparison of CHF values

Fluid	ΔT (ºC)	Berenson`s correlation [W/cm^2]	Son and Dhir [W/cm^2] (1997)	Present work [W/cm^2]
Water	100	2.14	1.21	--
Water	110	2.29	--	0.42
Water	130	2.60	--	0.83
Water	158	3.01	2.06	--

Table 4 Comparison of heat flux for film boiling

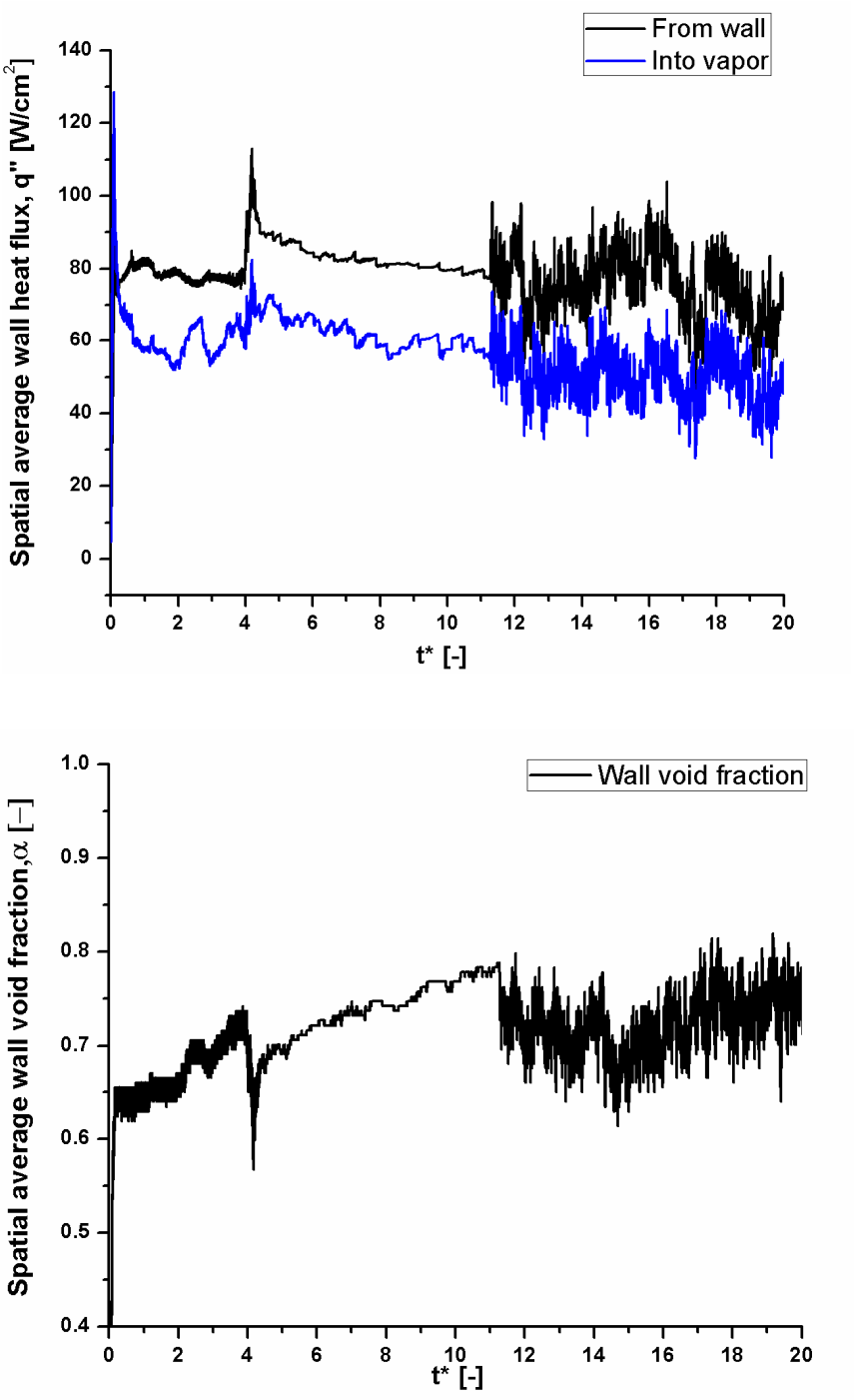


Figure 17 Temporal variation of heat fluxes and wall void fraction (CHF, ΔT = 27.5 ºC)

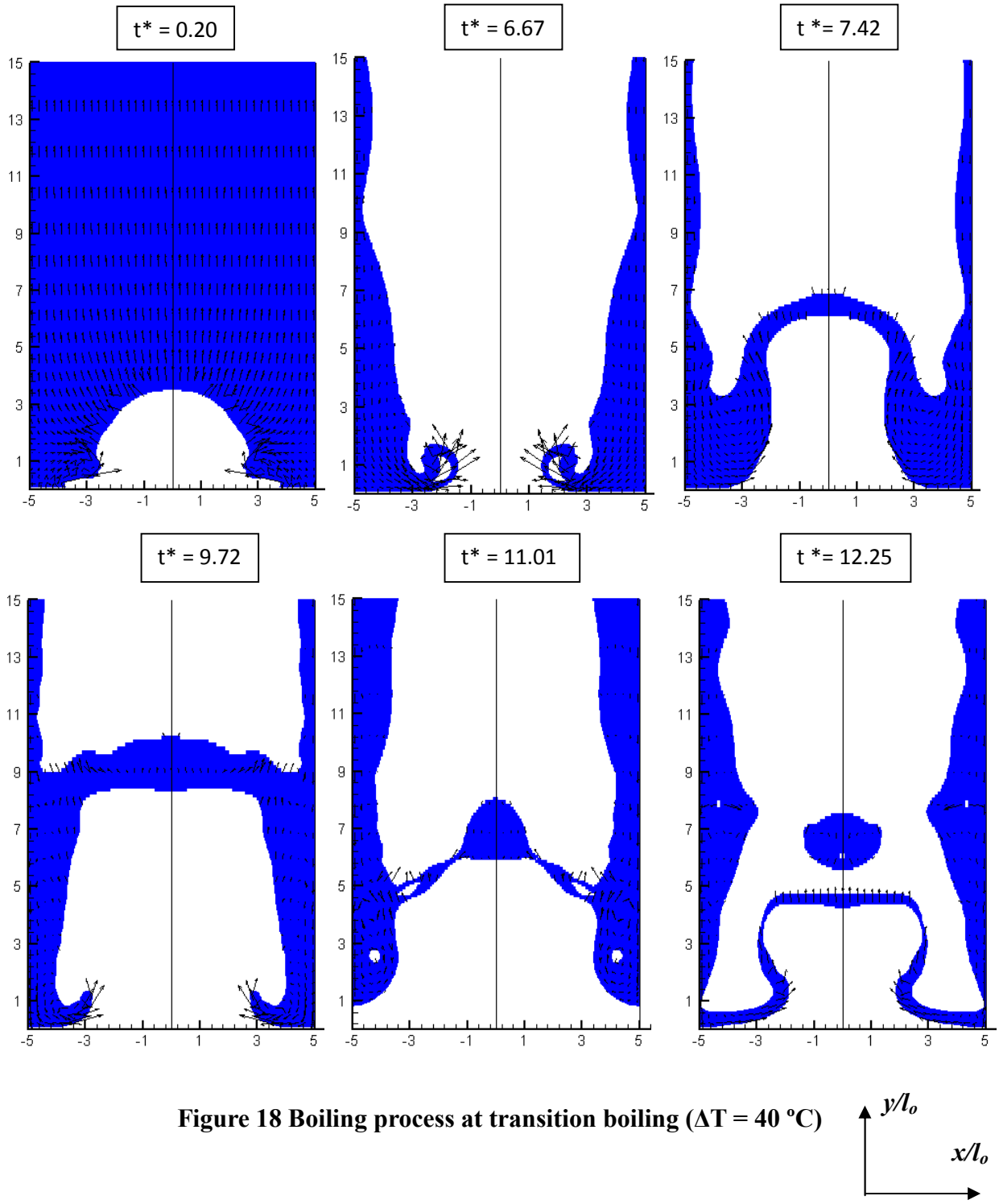


Figure 18 Boiling process at transition boiling (ΔT = 40 ºC)

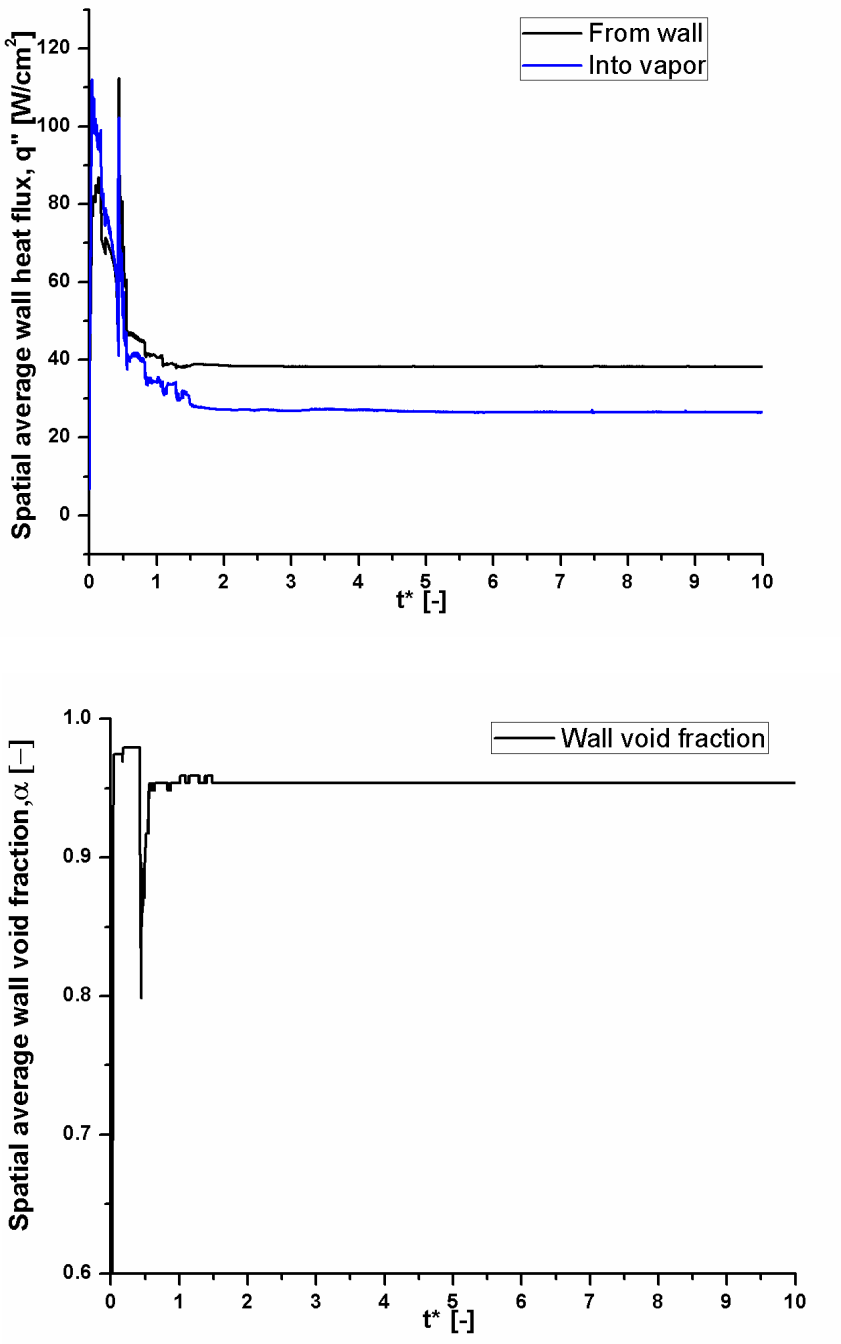


Figure 19 Temporal variation of heat fluxes and wall void fraction (Transition boiling, ΔT = 40 ºC)

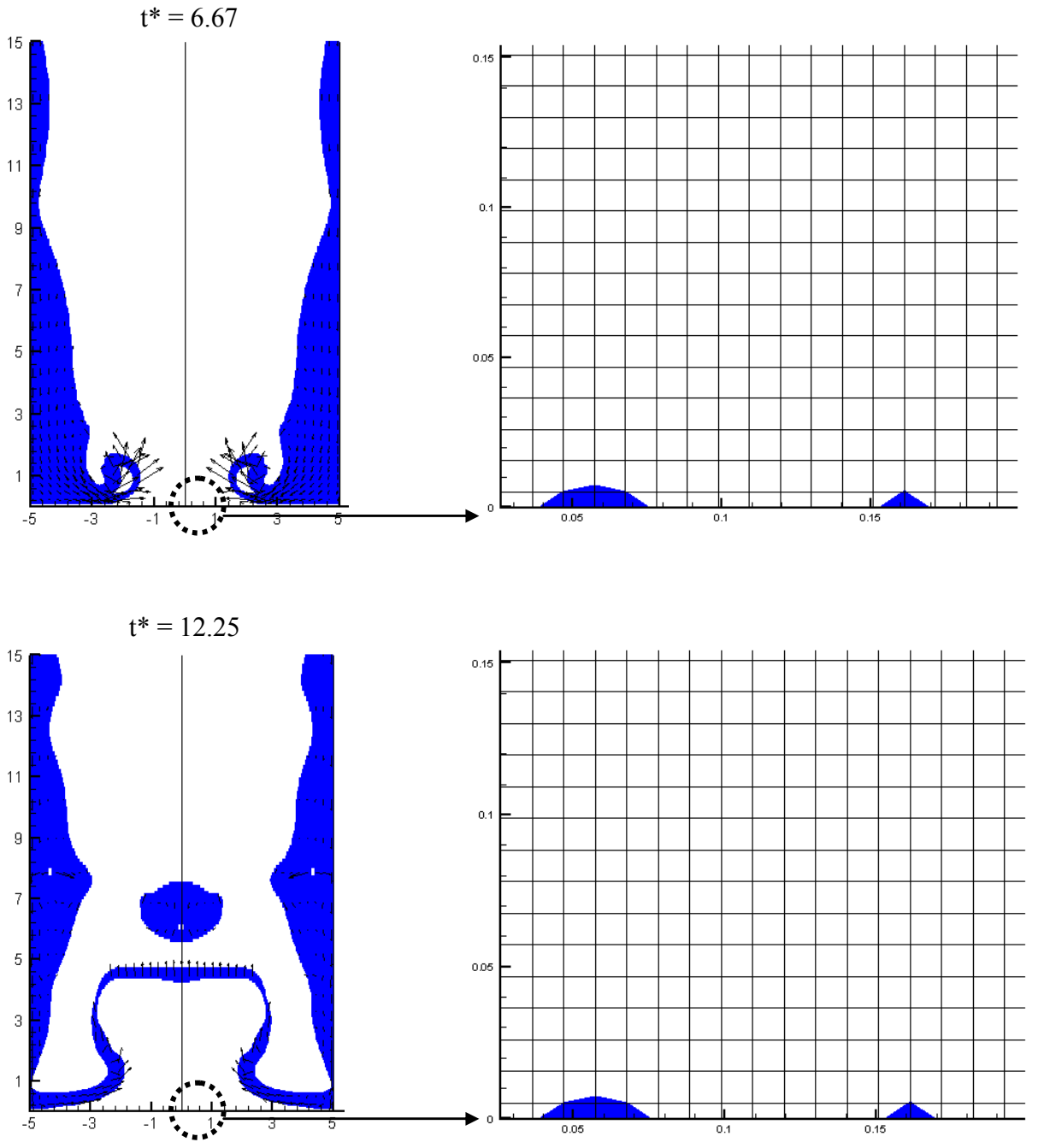


Figure 20 - Shape of the vapor stem at the surface

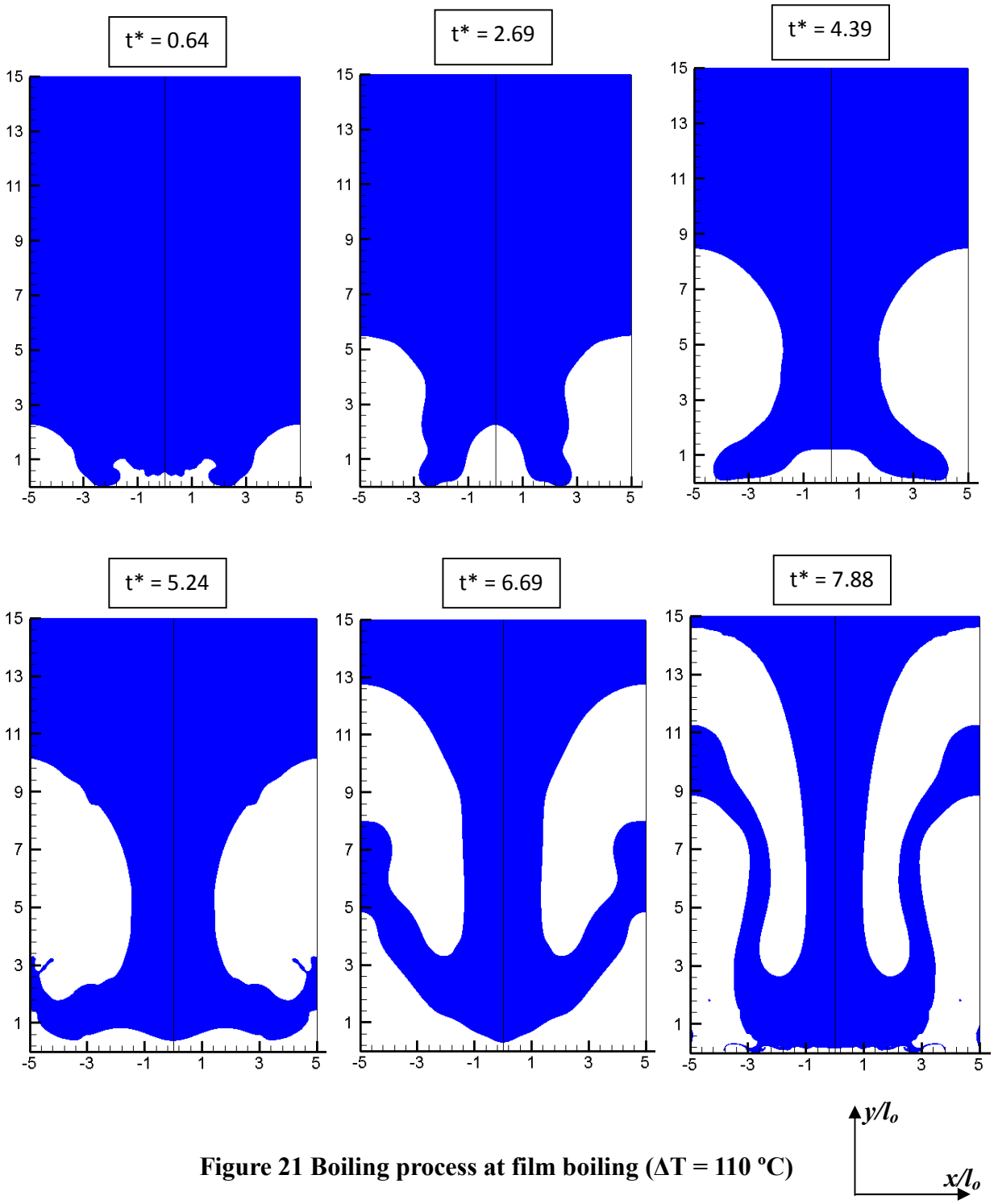


Figure 21 Boiling process at film boiling (ΔT = 110 ºC)

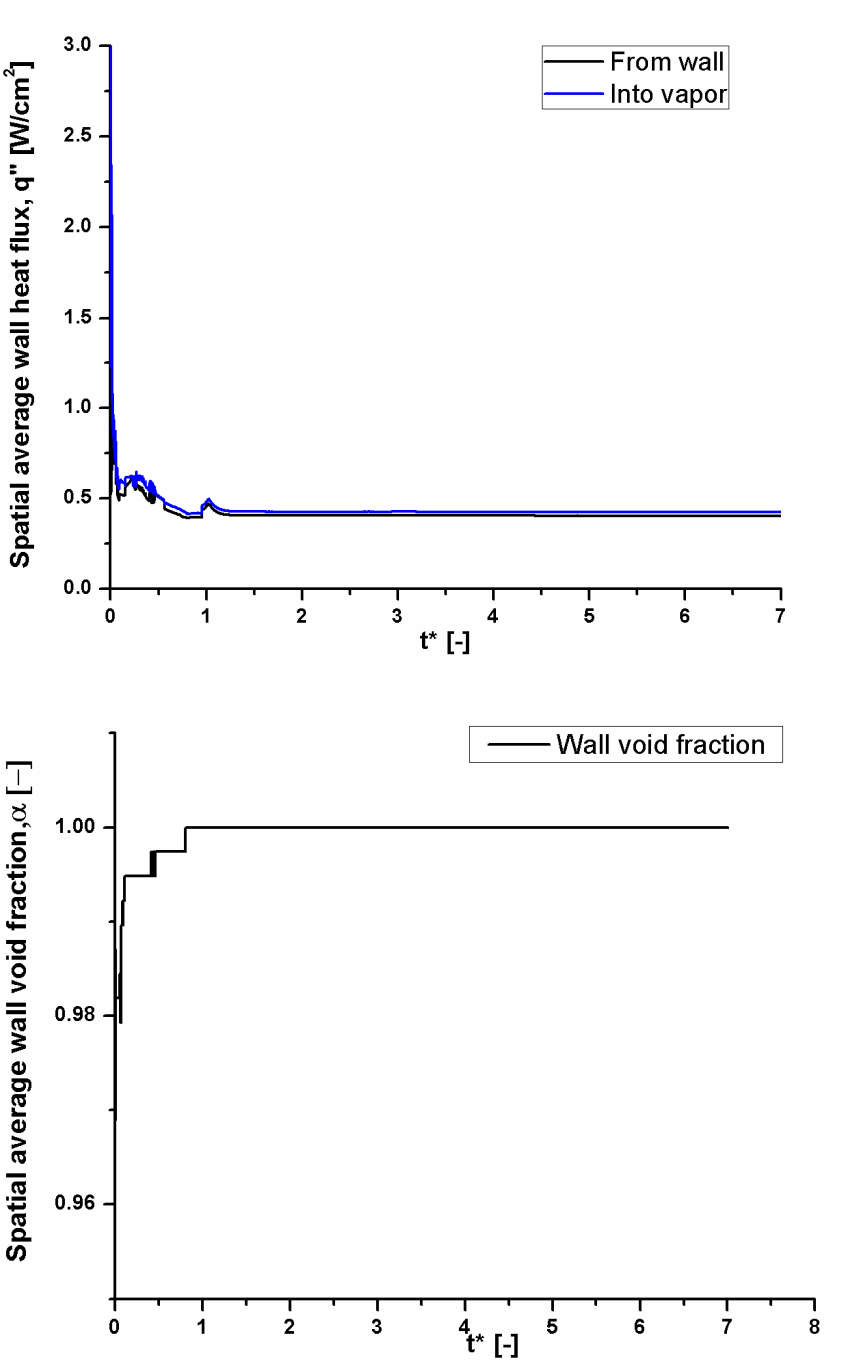


Figure 22 Temporal variation of heat fluxes and wall void fraction (Film boiling, ΔT = 110 ºC)

At a wall superheat of 110 ºC the boiling process is in the regime of film boiling as shown in Figure 21. Only the first cycle of bubble evolution has been shown here. At t* = 4.39 full grown vapor bubbles have evolved and are about to detach from the vapor film formed at the surface while at t* = 5.24 the vapor bubbles have detached completely from the vapor film and start rising upwards. The variation of heat fluxes and wall void fraction is plotted in Figure 22. Since the value of wall void fraction is unity everywhere the value of the computed heat flux from the wall and into the vapor are same.

At a wall superheat of 130 ºC (Point d in Figure 26) the boiling process is in the regime of film boiling as shown in Figure 23. Only the first cycle of bubble (which is two-dimensional in shape) evolution has been shown here. At t* = 4.81 thin vapor film is seen at the surface while at t* = 44.40 full grown vapor bubbles have evolved and are about to detach from the vapor film formed at the surface. The second set of bubble is formed at the center at t* = 64.13. The variation of heat fluxes and wall void fraction is plotted in Figure 24 showing the bubble release points also. Since the value of wall void fraction is unity everywhere the value of the computed heat flux from the wall and into the vapor are same. It has to be noted that the initial instability seen at the surface is captured from the numerical model naturally otherwise an initial specification of the vapor film profile has to be user defined in some cases (FLUENT tutorial: Horizontal film boiling, 2005).

Figure 25 shows the fluctuating interface velocity at the middle of the surface. At approximately t* = 40 is close to zero as bubbles are about to be released at the edges of the surface (x* = -5 and x* = 5). The peaks at t* = 50 and t* = 65 corresponds to the velocity of the departed bubble for the case of ΔT = 130 °C and ΔT = 170 °C. Temporal and spatial averaged wall heat flux

and wall void fraction are plotted versus wall superheat neglecting the variations during early periods of computation ($t^* \leq 40$). Figure 26 shows the simulated boiling curve compared against experimental data obtained Gaertner, 1965 (horizontal surface, unspecified contact angle) and Stephan and Abdelsalam correlation (1980) for the nucleate boiling regime and with Berenson`s correlation for the film boiling regime.

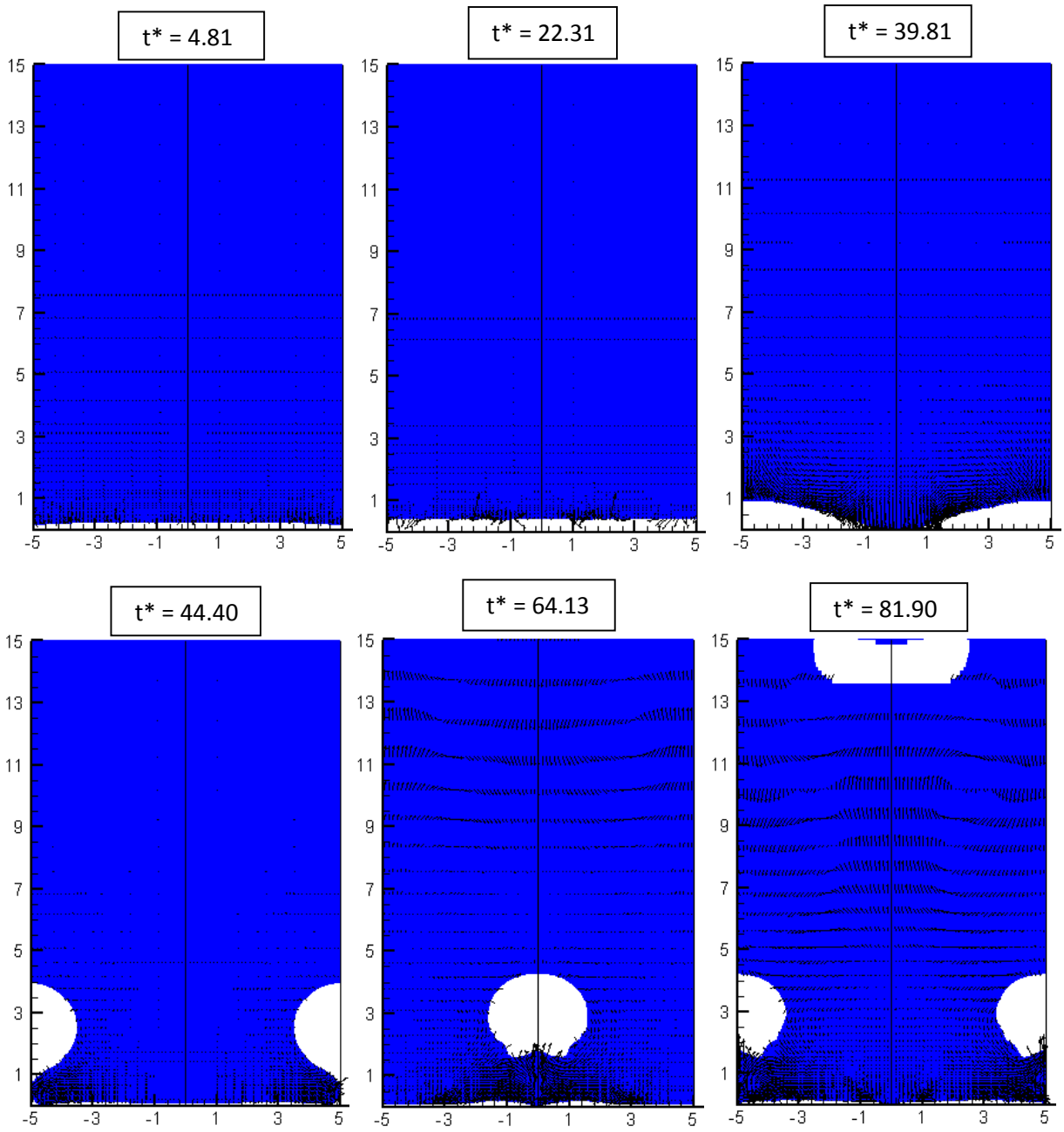


Figure 23 Boiling process at film boiling (ΔT = 130 ºC)

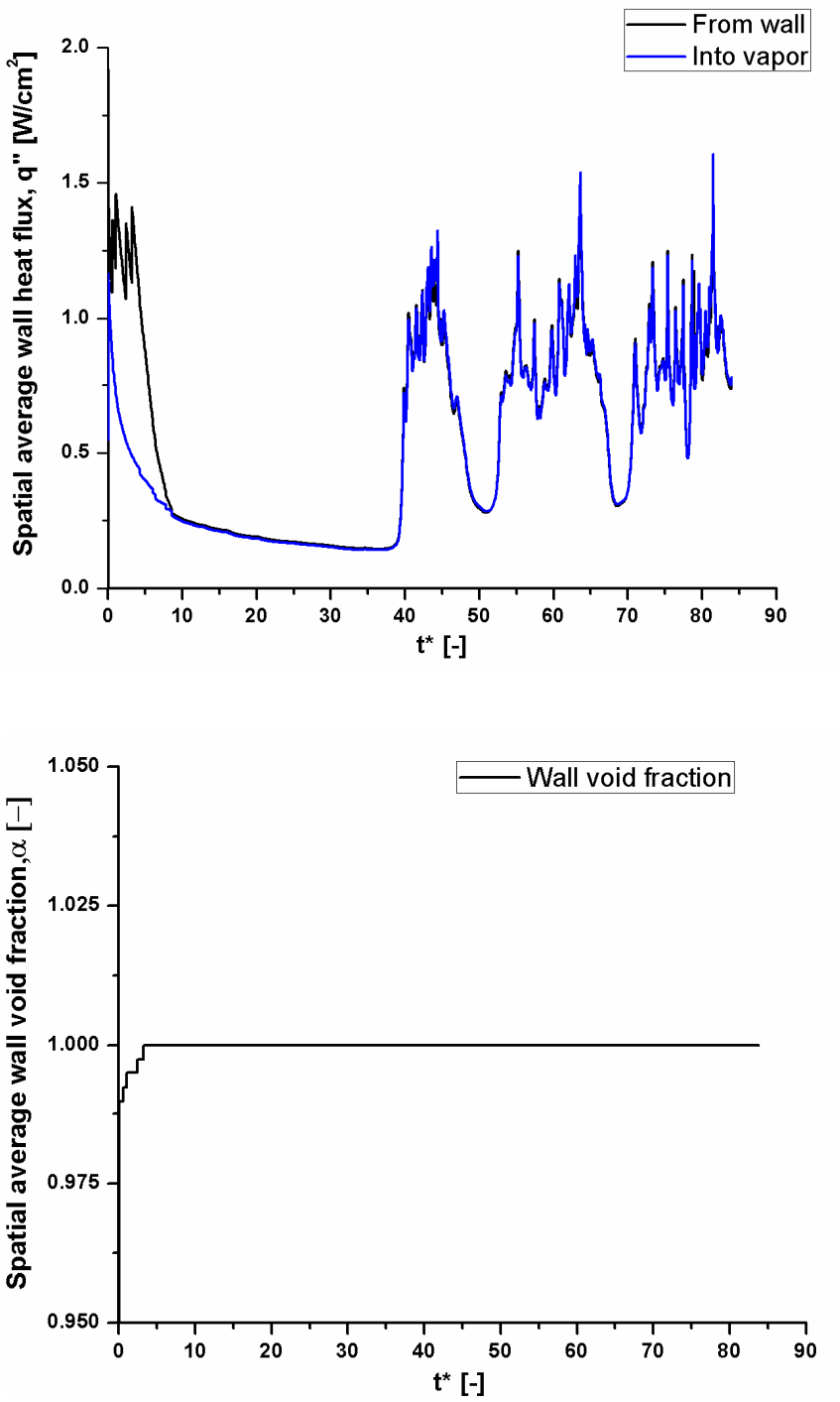


Figure 24 Temporal variation of heat fluxes and wall void fraction (Film boiling, ΔT = 130 ºC)

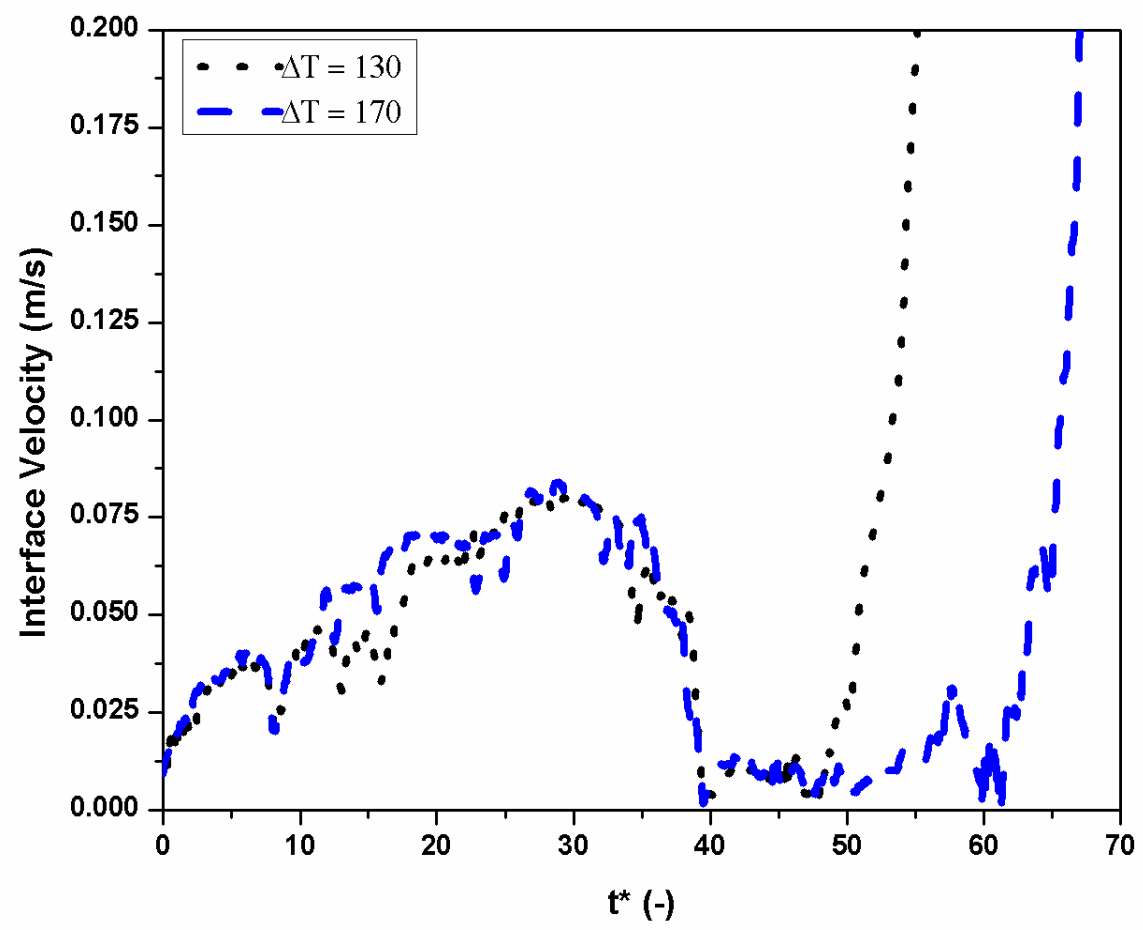


Figure 25 Interface velocity at the centre during the early evolution period (φ=38º, ΔT = 130 ºC & 170 ºC)

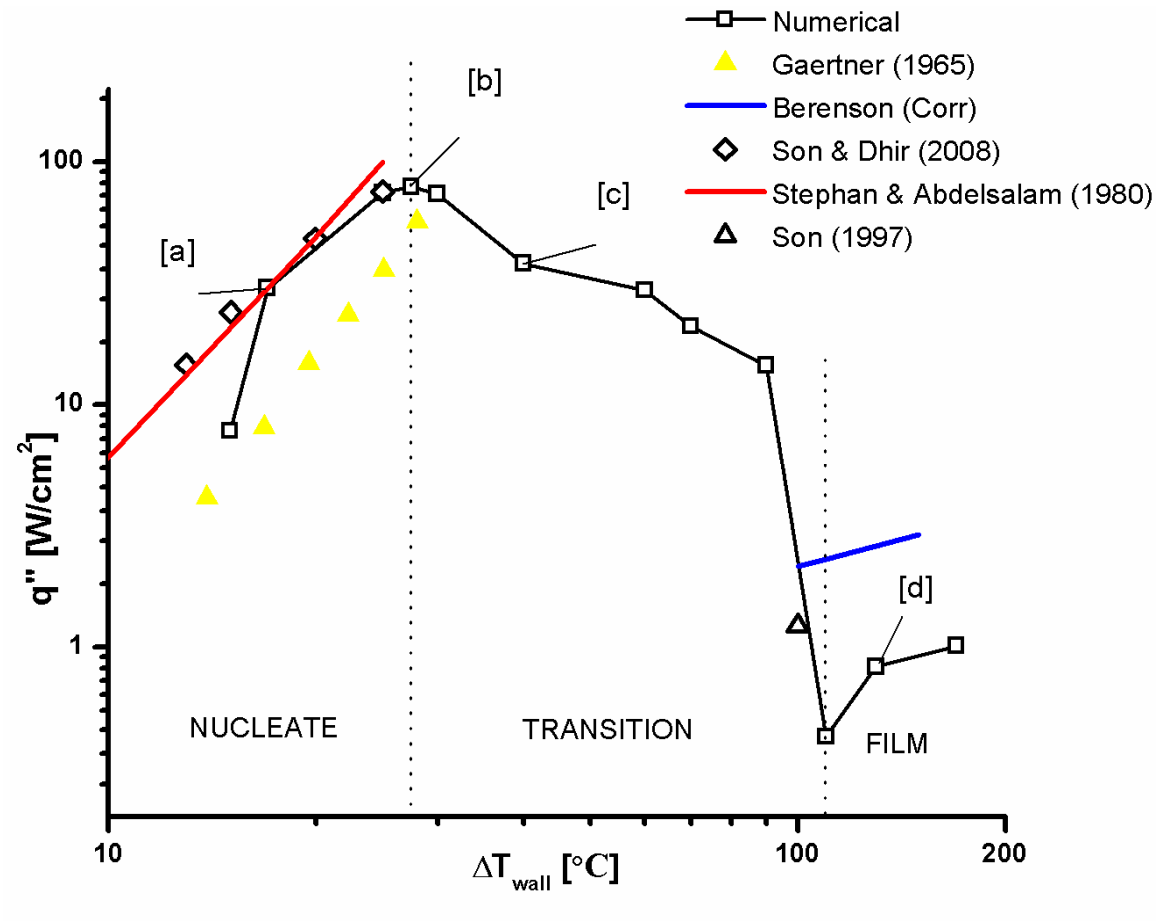


Figure 26 Boiling curve for the case of contact angle of 38°

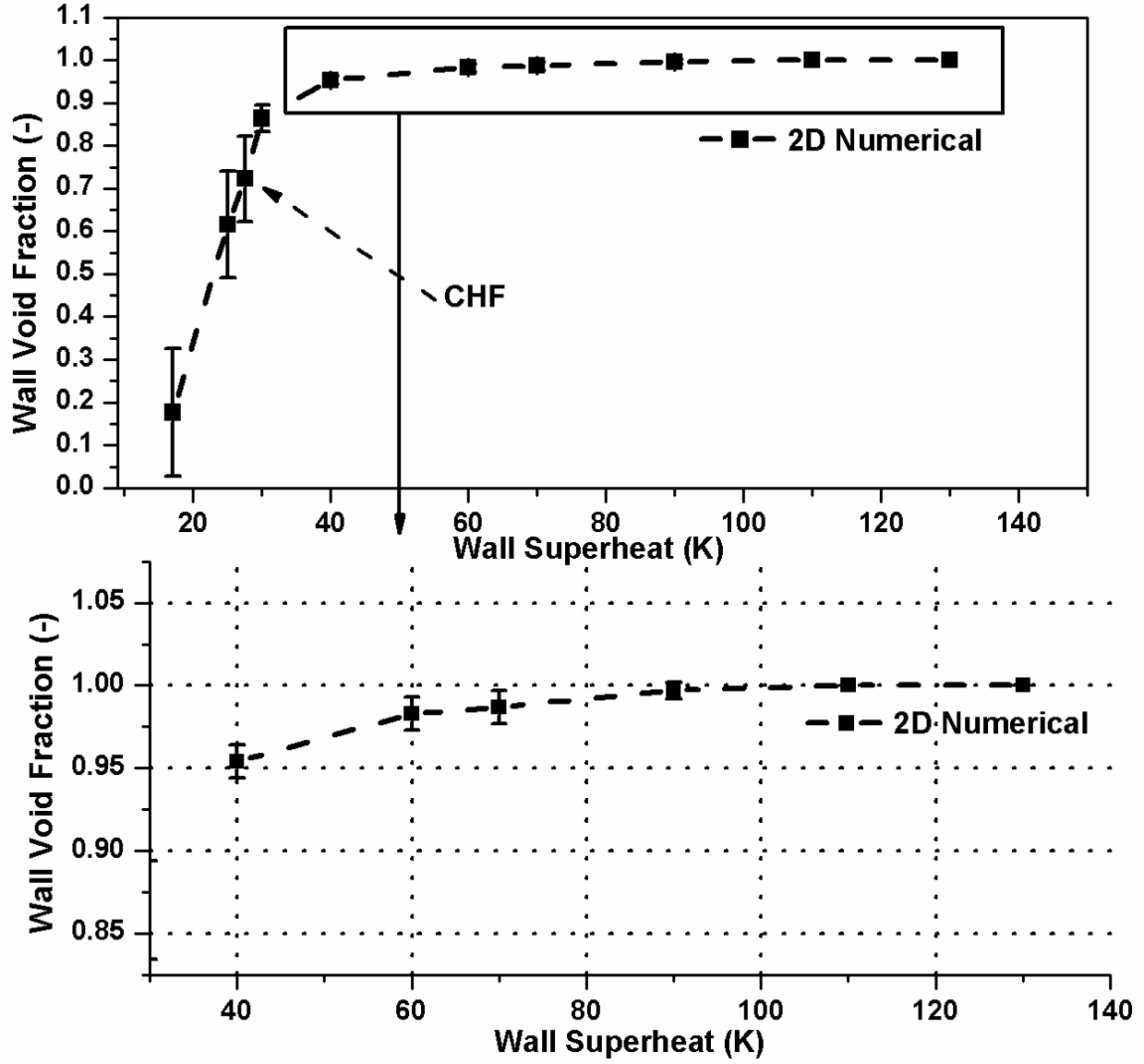


Figure 27 Variation of wall void fraction for contact angle of 38º

4.2 Results and Discussion - Variable Contact Angles (φ = 27°, 54°, 69°, 90°)

Figure 29 shows the simulated boiling curves obtained by extending the current numerical model for different contact angles. Even though time dependent conservation equations are being solved, the results are shown only when quasi-static conditions are reached in terms of stabilization of wall heat flux and wall void fraction. The effect of change in contact angle on the boiling curve is clear as the contact angle decreases the boiling curve shifts towards right. Major discrepancy is seen in the trend of the nucleate boiling regimes for the set of simulated boiling curves in the sense that with an increase in contact angle, the nucleate boiling heat flux is lower in comparison to the lesser contact angle case. For eg., at a wall superheat of 18 °C, the nucleate boiling heat flux of 90° is lowest while 27° is highest. Possible explanation for this incorrect trend is that the two-dimensional approximation manifests itself as horizontal vapor cylinders rather than spherical vapor bubbles in reality. For all the contact angles maximum heat flux computed is less than the hydrodynamic theory with the deviation from the hydrodynamic theory yielded values being more as the wettability of the surface is decreased. It is also seen that the superheat at which the maximum heat flux occurs rises as the contact angle is decreased, and thus it can be inferred that the slope of the locus of q_{max} in Figure 29 is around 3 with respect to wall superheat.

The overall heat transfer rate from the surface is the summation of heat removal from both the dry and the wet areas on the surface. In the present analysis, the surface temperature over both the dry and wet areas is held constant which is realizable for thick heaters with high thermal conductivity as for heaters which have poor thermal conductivity and are thin, a significant difference in surface temperature might exist (Dhir (1998)). Hysteresis of the boiling curve is not

considered in the present study but as reported by Auracher *et al.* (2004) that single and reproducible boiling curves are obtained provided if precise temperature control system is used in conjunction with a clean heating surface, as even minimal deposits on the surface has the tendency to shift the boiling curve with successive runs. They also stated that the situation is however different for transient modes as the heating and cooling transients yield different boiling curve even for clean surface conditions.

Figure 28 shows the steady state maximum heat flux obtained from the current numerical model. As stated earlier the CHF is seen decreasing with increase in contact angle and the slope of the locus of CHF with variation in contact angle is negative with a magnitude of 1.2 approximately. The numerical value of CHF obtained from the present simulations are in good agreement to the experimental data reported by Hahne and Diesselhorst (1978) done on horizontal cylinders. The data obtained by Liaw and Dhir (1989) is for vertical surface and it can be inferred by virtue of its slope that the effect of contact angle on CHF is weak for the vertical surface as compared to the horizontal surface. For simulations with contact angle of 90° the microlayer contribution is not accounted for with the assumption that there is no microlayer formation. Subsequently, it was found out that the CHF value for a contact angle of 90° is only 8% of that given by Zuber (1959) even when 20% of the surface is covered with liquid. This reinforces the reasoning provided by Liaw and Dhir (1989) that for poorly wetted surface the maximum heat flux conditions are limited by the rate of evaporation near the surface as surface still has access to liquid.

Transition boiling curve is characterized by the negative slope on the boiling curve due to the reduction in wall heat flux on increasing the wall superheat and is inherently unstable.

Transition boiling curve also depends on whether it is being approached from nucleate boiling or film boiling side. The effect of surface wettability on hysteresis on the transition boiling curve has been investigated by Liaw and Dhir (1986). Their experiments have shown that the transition boiling heat fluxes during heating mode is higher than that obtained from the cooling mode and the difference in the transition boiling curves obtained from the cooling mode and the heating mode diminishes as the surface wettability reduces. Another study done by Maracy and Winterton (1988) on horizontal surface using water also showed hysteresis at lower contact angles as well with the transition boiling heat flux being higher in the heating mode than in the cooling mode. In the present study the simulations are started from bubble inception rather than using the converged solution of lower wall superheat simulation result as the initial condition.

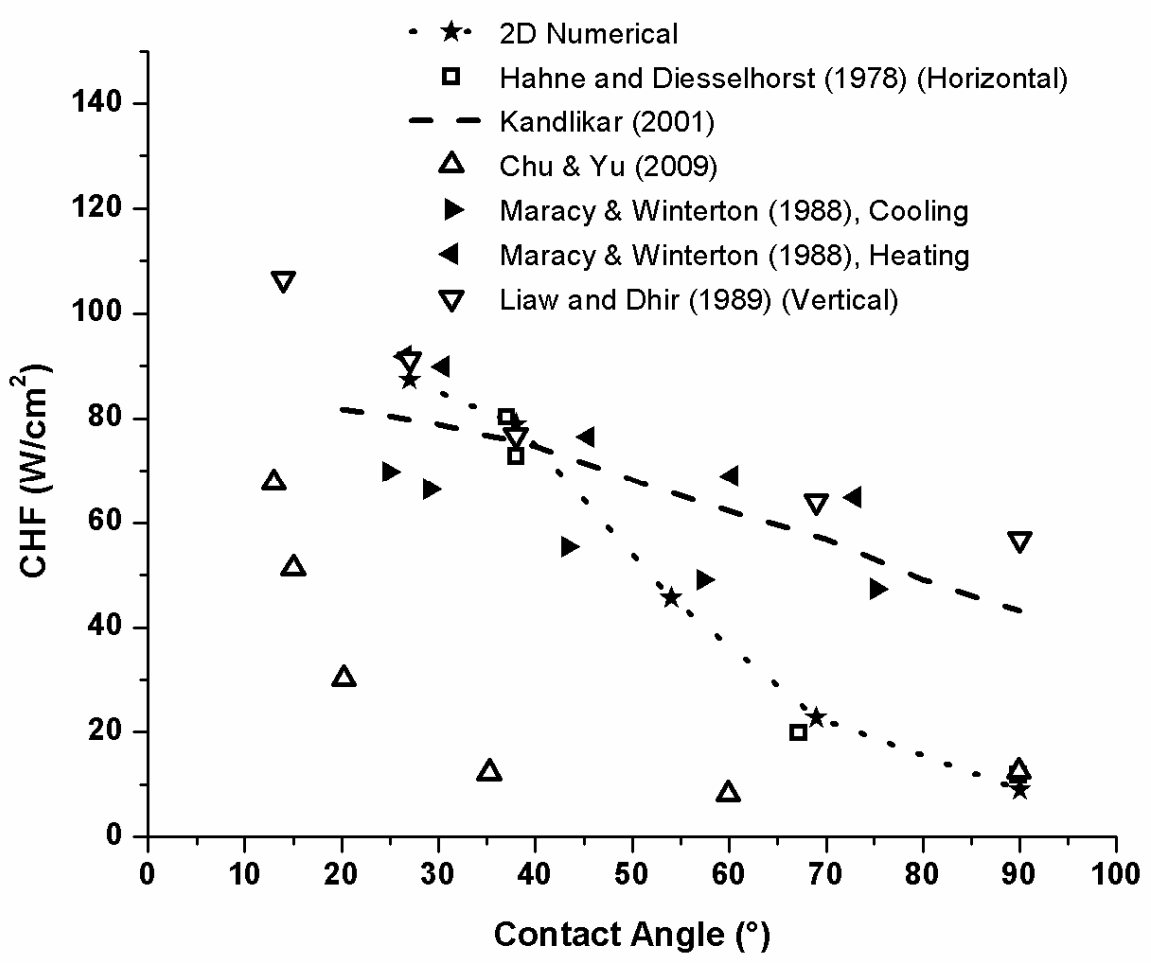


Figure 28 Comparison of influence of contact angle on CHF

Figure **30** shows the variation of wall void fraction with superheat for different contact angles. Computation of wall void fraction is vital, especially in the transition boiling regime which is unstable by virtue of it being a combination of unstable nucleate boiling and unstable film boiling. Some of the models applicable for the transition boiling models rely on the fraction of liquid contact time on the heater surface. For instance, it has been suggested by Bjornard and Griffith (1977) that the heat flux at any wall superheat is the combination of maximum heat flux (q_{max}) and minimum heat flux (q_{min}) and hence can be given by:

$$q = F_L q_{\max} + (1 - F_L) q_{\min} \tag{86}$$

where, F_L is the fractional area occupied by the liquid and is correlated by:

$$F_L = \left[\frac{T_{\min} - \mathrm{T}}{T_{\min} - T_{\max}} \right]^2 \tag{87}$$

Another way to estimate transition boiling heat flux was proposed by Liaw and Dhir (1986), in which the transition boiling heat transfer coefficients were expressed as the combination of the heat transfer coefficients over wet and dry areas as given below:

$$h = \beta_{h,c} F_l h_l + h_g (1 - \beta_{h,c} F_l) \tag{88}$$

In the above expression, h_l and h_g are time averaged heat transfer coefficients over the wet and dry areas respectively, F_l is the time averaged fractional area of the heater accessible to the

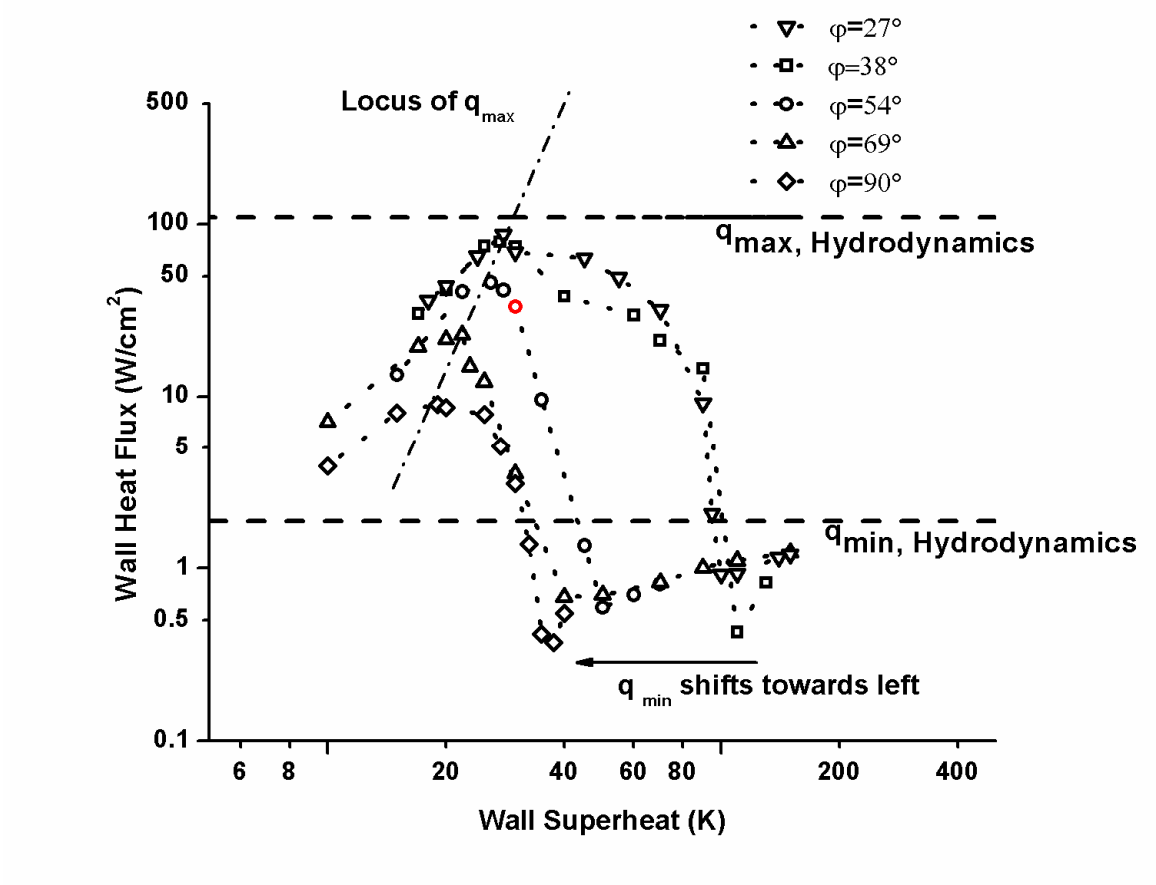


Figure 29 Comparison of numerically simulated boiling curves for different contact angles

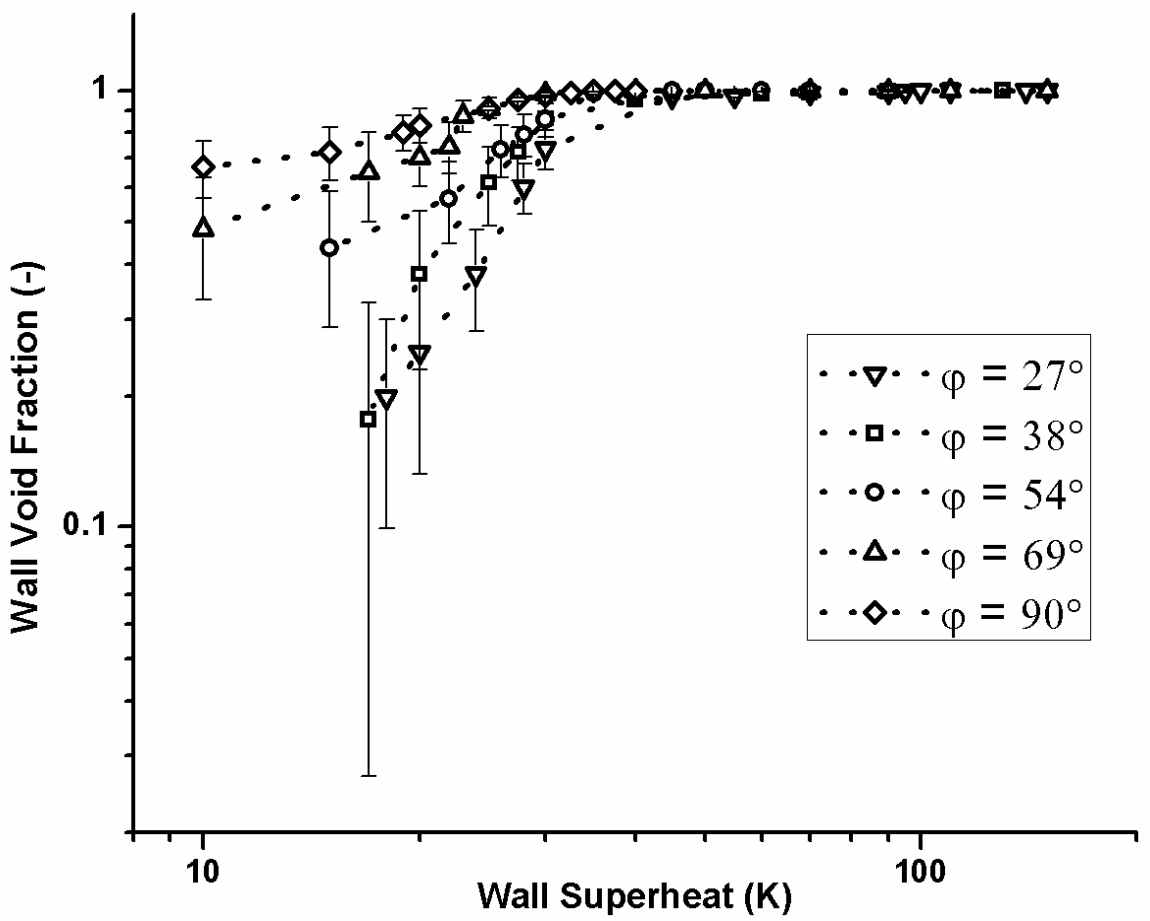


Figure 30 Comparison of numerically computed wall void fraction for different contact angles

liquid, $\beta_{h,c}$ is a parameter depending on heating curve or cooling curve with its value decreasing with increase in contact angle and lies between 0 to 1.

Figure 31 shows the variation of minimum heat flux, q_{min} ,as a function of contact angle. It can be seen that the numerically simulated values are lower than those obtained by Liaw and Dhir (1989) for experiments in heating mode on a vertical plate, and are 50-75% less than those given by Berenson`s correlation. Overall, in comparison to both experimental and theoretical values very low heat flux is seen in the film boiling region due to vapor fractional area close to unity. The q_{min} values from simulation decreases linearly approximately by a factor of 2 as the contact angle is increased from 27° to 90° since the superheat at which q_{min} occurs is less for the latter with near about same void fraction. This is in agreement to the reasoning provided by Liaw and Dhir (1989) that the incoming liquid spreads quickly on a well wetted surface and hence transition boiling starts earlier with q_{min} being higher.

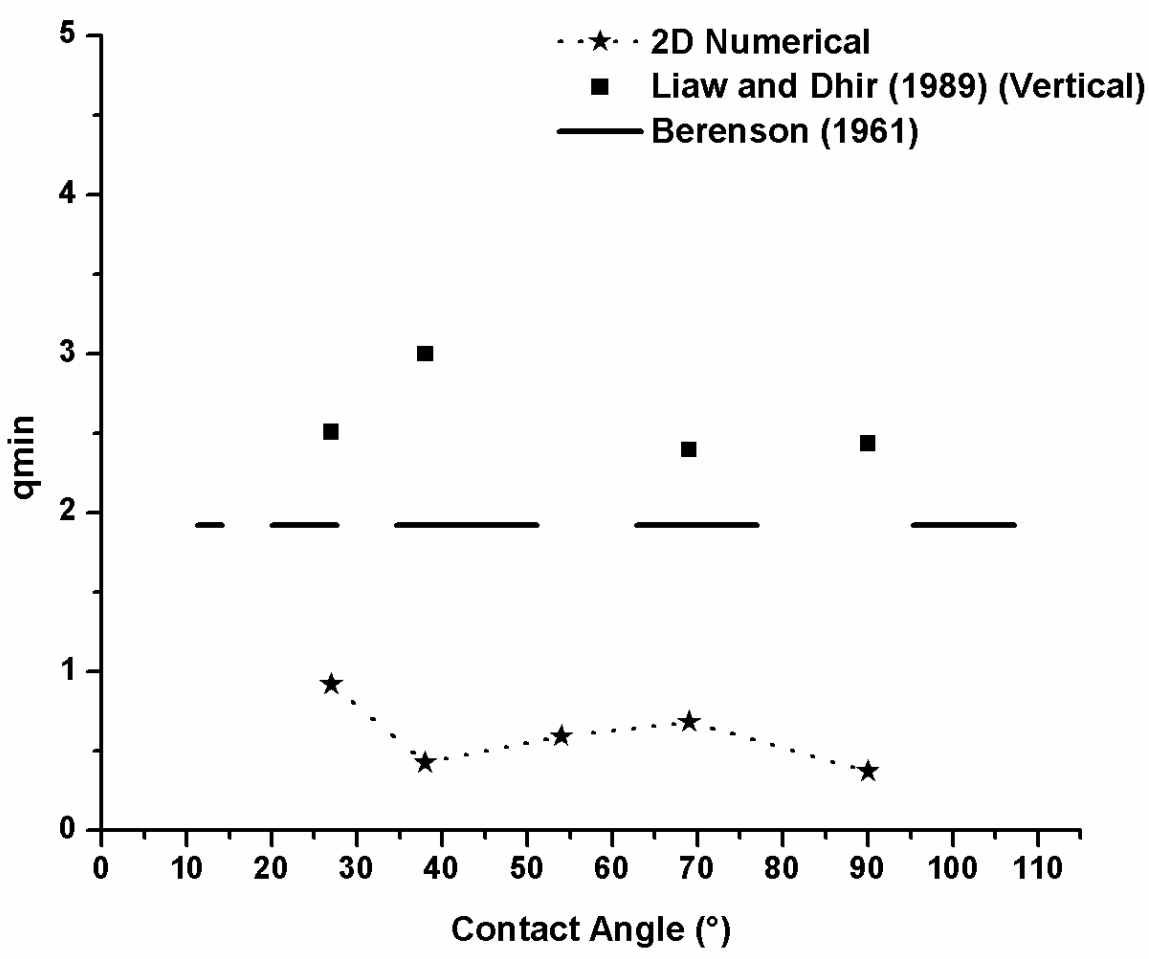


Figure 31 Variation of minimum heat flux for different contact angles.

4.4 Conclusion

This chapter was focused on attempting to simulate boiling curves for different contact angles under the assumption that boiling process is two-dimensional. The nucleation site density for the two dimensional was obtained by conversion from the correlation values for three dimensional case. The values of q_{max} and q_{min} decreases with increase in contact angle indicating that the trend is consistent with literature. However the trend seen in the nucleate boiling heat flux with contact angle at a particular wall superheat (before CHF) was completely opposite to the trend reported in the literature as the two-dimensional approximation makes a symmetric boundary condition approximation in the third dimension, and hence manifesting an actual spherical vapor bubble to infinitely long cylindrical vapor structures. This incapacitates the two-dimensional model to yield any meaningful physical results and reinforces the statement that three-dimensional simulation are the only way to numerically study the boiling process.

5. Three Dimensional Boiling Curve Simulation

This chapter focuses on carrying out 3D simulations of boiling process. Since 3D simulations require tremendous amount of computational time so both coarse grid results with long run times and fine grid simulations with short run time have been performed. Even though fine grid results are more accurate than coarse grid results *when converged*, the coarse grid results have an excellent property that residuals drop very rapidly as shown in Figure 32. The computational areas for coarse grids and fine grids are taken as dimensionless length and width as 5.5 x 5.5 (1.89 cm^2, 66 x 98 x 66) and 2 x 2 (0.258 cm^2, 98 x 194 x 98) respectively. The domain dimensions for coarse grid case has been selected to accommodate Taylor`s most dangerous wavelength while for the fine grid cases is smaller as the intention is to investigate the effect of grid size on computational results.

5.1 3D Coarse Grid Simulations, Case I (φ=38°)

Three dimensional numerical simulations on coarse grids were carried out using the properties of saturated water at 1 atm. The contact angle was considered to be static with a value of 38º. In reality, the contact angle varies dynamically between an advancing (maximum) contact angle and a receding (minimum) contact angle depending on the speed of the contact line reviewed earlier by Dusan (1979). Since, the value of non-dimensionalized Taylor`s most dangerous wavelength ≈ 5, the non-dimensionalized computational domain length and width for the present study is chosen to be $0 \leq x \leq 5.5$, $0 \leq y \leq 11$ and $0 \leq z \leq 5.5$ resolved with a grid density of 66 x 98 x 66.

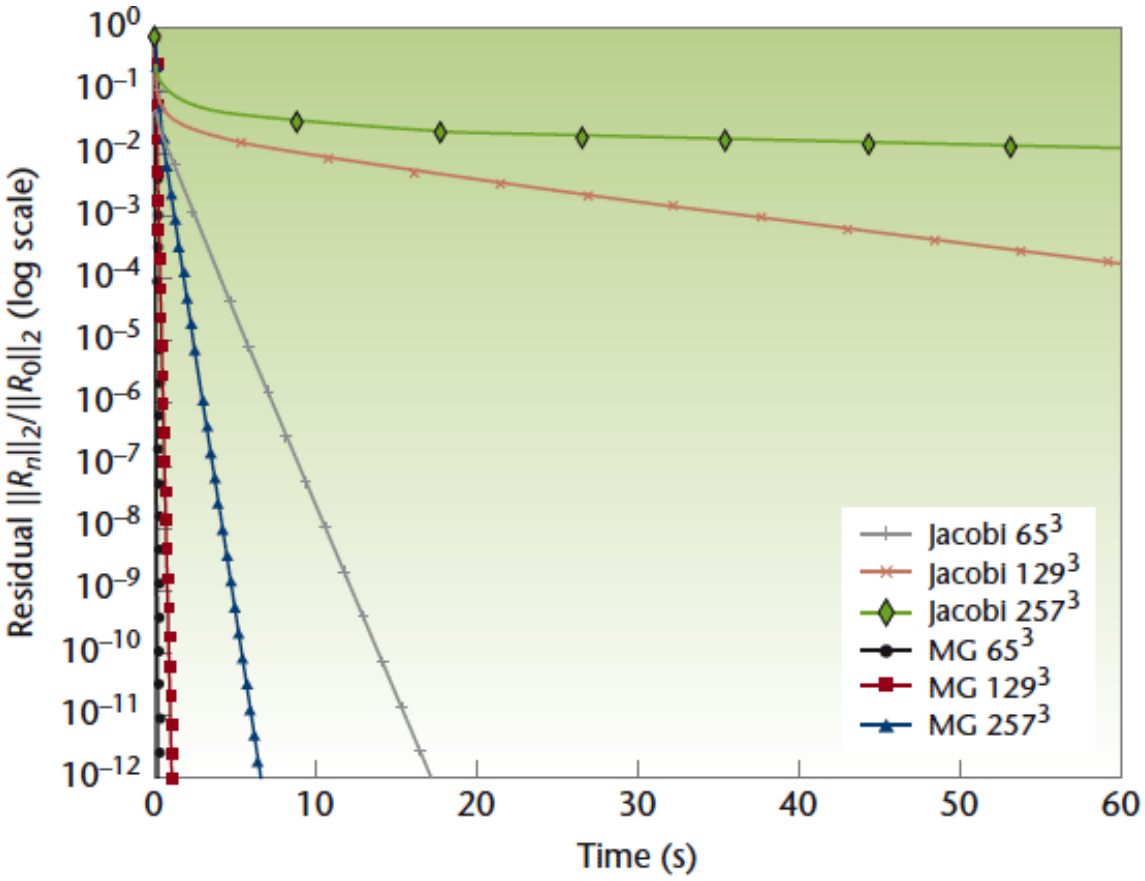


Figure 32 Residual reduction on different grids (Jacobsen *et al.* (2013))

Wall superheat, ΔT, (°C)	Cavity density, (Basu *et al.*, 2005) φ = 38° Na , [Sites/cm^2]	Number of cavities, N φ = 38° (3D Coarse) Area = 1.89 cm^2	Number of cavities, N φ = 38° (3D Fine) Area = 0.258 cm^2	Number of cavities, N φ = 69° (3D Fine) Area = 0.258 cm^2	Waiting time (Milliseconds)
15	12	24	4	9	2.09
18	32	61	8	24	0.64
20	57	107	15	44	0.43
24	149	281	38	116	0.25
28	337	400	87	263	0.18
33.5	871	400	225	680	0.12
40	2230	400	575	900	0.037
60	19124	400	900	900	0.007
100	286661	400	900	900	0.0008
125	935388	400	900	900	0.0003
130	1151512	400	900	900	0.0002

Table 5 Nucleation sites and waiting time used for the 3D simulations

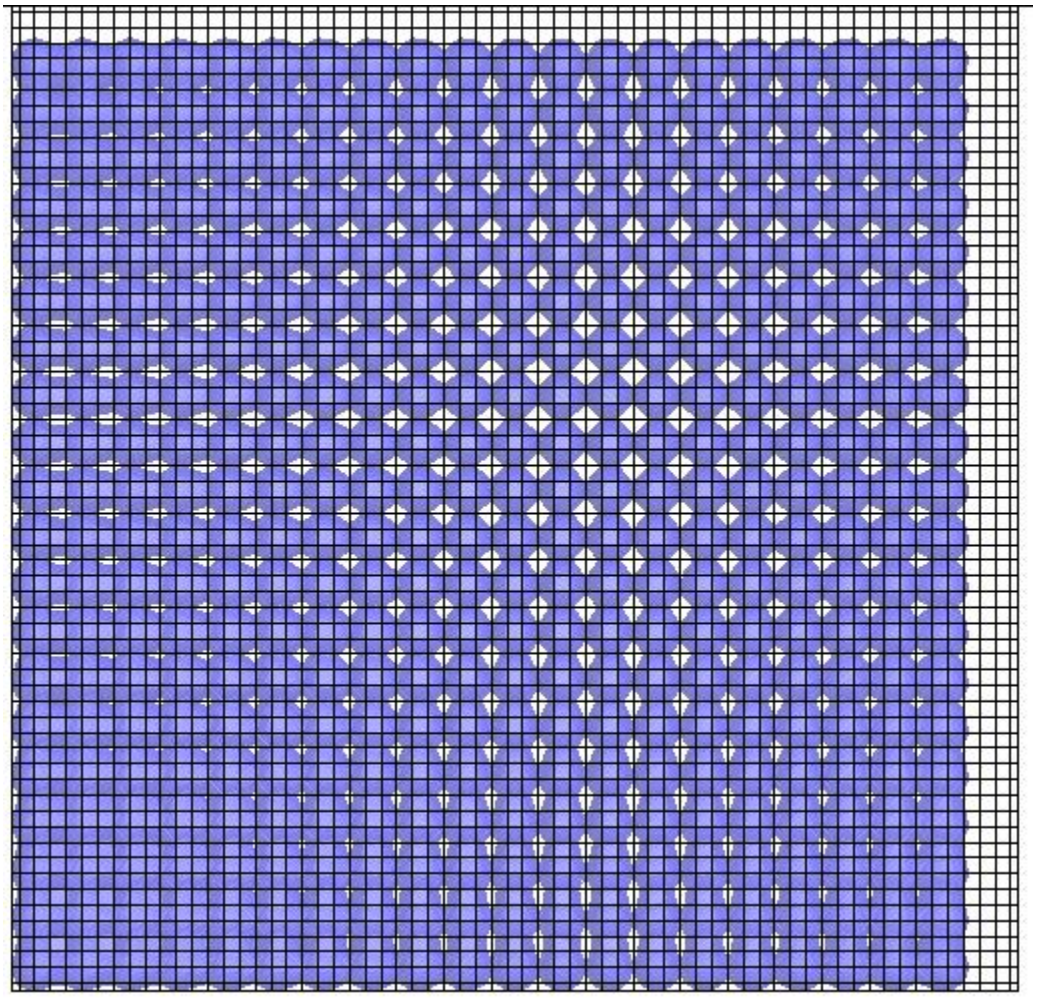
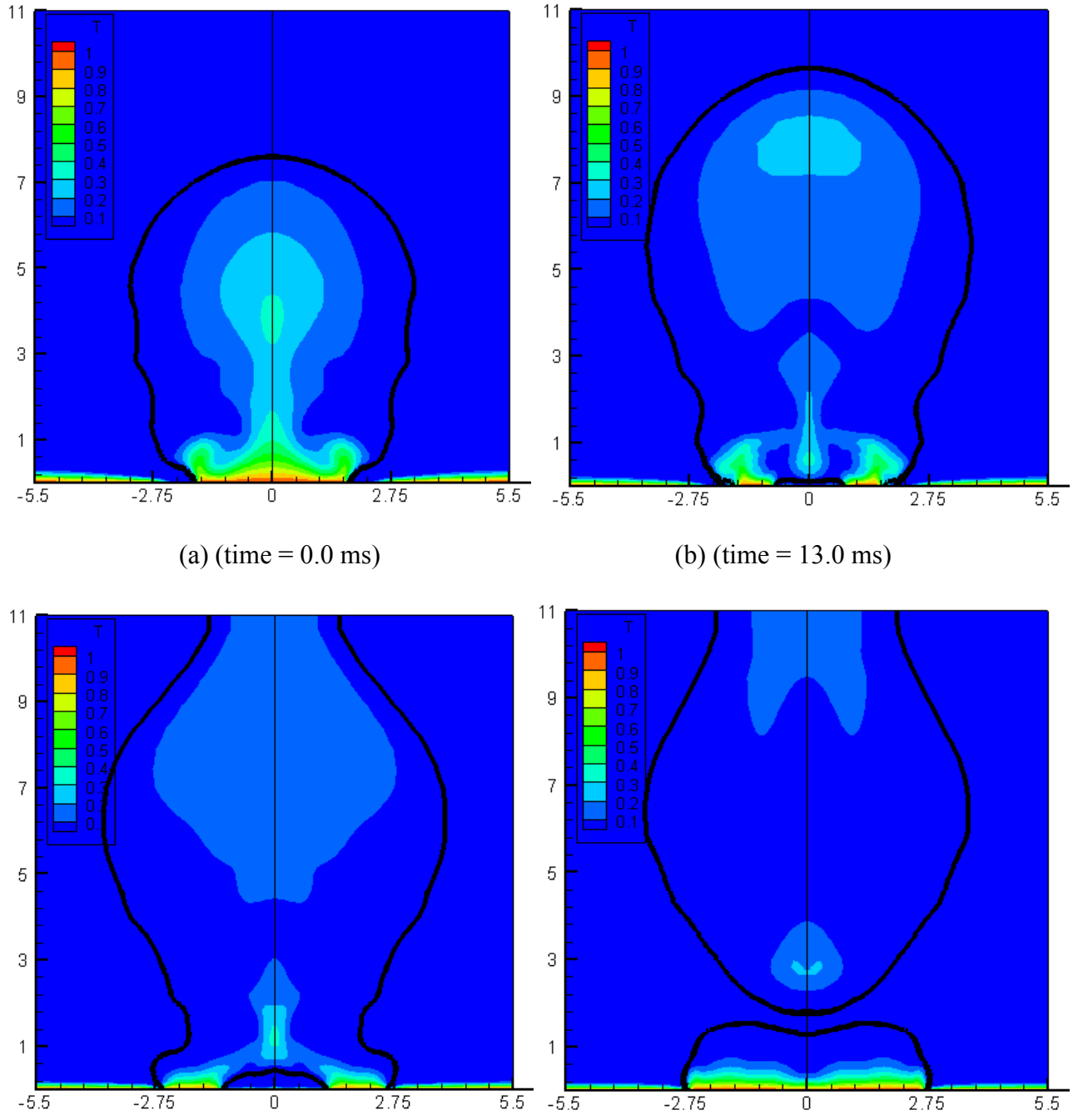


Figure 33 Maximum cavity distribution of 400 for coarse grid cases

Gaertner (1963) and Sultan and Judd (1978) found that active sites are randomly distributed on a heated surface obeying Poisson distribution. Also as pointed out in Shoji (2003), nucleation sites can interact with each other altering the local wall superheat and other parameters that determine the nuclei stability. They also affect the number of active sites causing intermittence in bubble production. However, for the present case small bubble embryos are placed sequentially along the heater surface with atleast 3-4 grid spacing apart and already nucleated at the start of the simulation. The addition of new bubble on a particular cavity occurs after the previous bubble has left with time interval being equal to the waiting time. Table 5 shows the number of cavities and the waiting time obtained from the correlation of Basu *et al.* (2005) for different superheats considered for the current work. The maximum cavities that can be placed depends on the number of grids and hence the fine grid cases can accommodate more cavities than the coarse grid cases. It has to be noted here that upon successive nucleation the creation of artificial vapor bubble displaces the already present superheated liquid thus altering the mass and energy content of the computational domain and manifests itself as the numerical artifact. Contrary to the two-dimensional case the vapor is not assumed to be saturated and hence energy equation is being solved for both the phases. Figure 34- Figure 37 shows the bubble evolution for different regimes of the boiling curve at a cross section of z = 3.49 mm which is at quarter length of the z extent of the domain. The starting bubble evolution period of 30 ms during which the mushroom shaped bubble is formed is



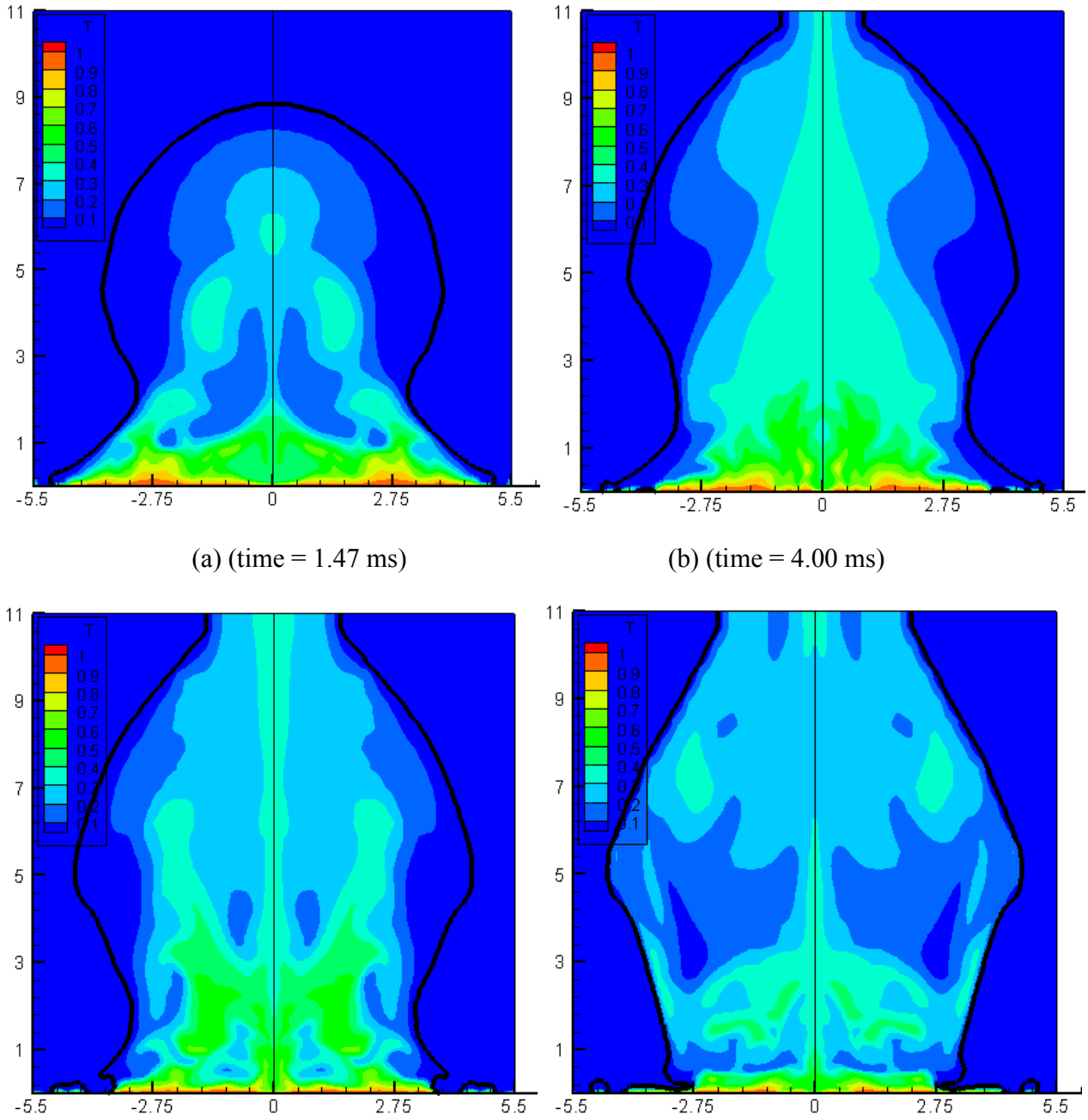
(a) (time = 0.0 ms)

(b) (time = 13.0 ms)

(c) (time = 22.3 ms)

(d) (time =30.2 ms)

Figure 34 Bubble evolution at nucleate boiling (Z=3.49 mm, ΔT=15 °C)



(a) (time = 1.47 ms)

(b) (time = 4.00 ms)

(c) (time = 4.94 ms)

(d) (time = 6.12 ms)

Figure 35 Bubble evolution at critical heat flux (Z=3.49 mm, ΔT=27 °C)

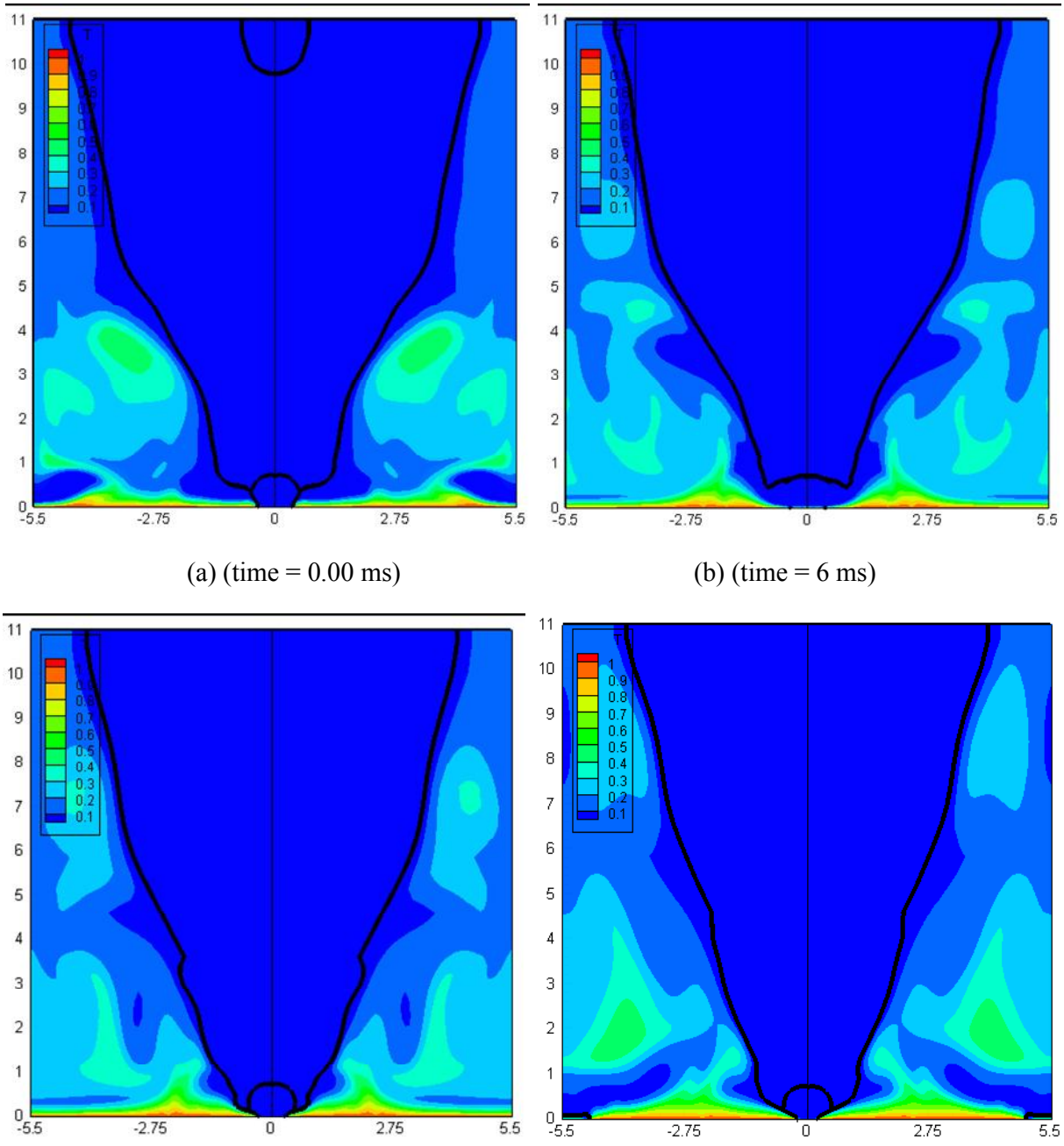
y/l_o

x/l_o

ignored and only the period for which the heat flux is stabilized demonstrating quasi-static case has been shown for Figure 34. The interface is shown as a dark black line separating the two phases while the temperature fields are shown as colored contours on a normalized scale of 0 (saturated) to 1 (wall superheat). Figure 34 (a), (b), (c) and (d) corresponds to t = 0.0 ms, 13.0 ms, 22.3 ms and 30.2 ms of Figure 38, showing time varying wall heat flux and wall void fraction. The vapor evolution configuration at a wall superheat of 15 °C is shown in Figure 34. Repeated formation of macrolayer, which is millimeter scale liquid layer left attached to the heater surface influencing nucleation and contact line phenomenon, beneath the mushroom bubble is seen. The mushroom bubble is being fed continuously by nucleating bubbles at the surface in addition to the vapor generated from the vapor stems implanted in the thermal layer (superheated liquid layer). From the temperature profile being developed close to the surface it can be seen that the thermal layer inside the vapor is elongating upwards due to the dominant effect of convection transport due to rising vapor while the thermal layer in the liquid side is constricted due to the downward inflow of the liquid from the top. The mushroom bubble formed breaks at a time of approximately 30 ms from the surface at which wall heat flux also drops as there are no formation of vapor stems beneath the bubble.

The area and temporal averaged wall heat flux for wall superheat of 15 °C is 15 [W/cm^2] approximately while wall void fraction being 0.15. Upon increasing the wall superheat to 27°C additional nucleation sites gets activated and a sustained vapor column is seen carrying superheated vapor out of the domain from the top. From Figure 39 it can be inferred that the average value of heat flux computed at CHF is about 60 [W/cm^2] with wall void fraction value being 0.68. The simulation from CHF onwards are done keeping the nucleation sites to a fixed

value of 400 as any more accumulation of nucleation sites is not possible from computational point of view (Figure 33) , as nucleating bubbles occupy 3-4 grid spacing and sites are denoted by point cavities located at discrete grid points. When a sustained superheat vapor column is formed (Figure 35), some of the nucleation sites are eclipsed by the vapor column base and only the sites which are away from the vapor columns nucleate. The bubbles thus generated drift inwards upon departure from the surface towards the vapor column under the action on liquid inflow at the sides. Upon increasing the wall superheat to 30°C the computed wall heat flux drops from its peak value of 60 [W/cm^2] to 40 [W/cm^2] marking the transition boiling regime as shown in Figure 36. Figure 37 shows the film boiling simulation observed at a wall superheat of 130°C. The entire surface is covered with a vapor film with the value of wall void fraction reaching unity. For the case of film boiling, no temperature gradient in the liquid phase is observed signifying zero liquid superheat. On the contrary, for the vapor phase the trend seen is opposite - for lower nucleate boiling regime the vapor superheat is less and as the wall superheat is increased the vapor gets more superheated. Similar observations were reported by Buchholz *et al.* (2006) who measured vapor superheat of approximately 2 [K] at lower nucleate boiling and upto 40 [K] vapor superheat for wall superheat of 80 [K].



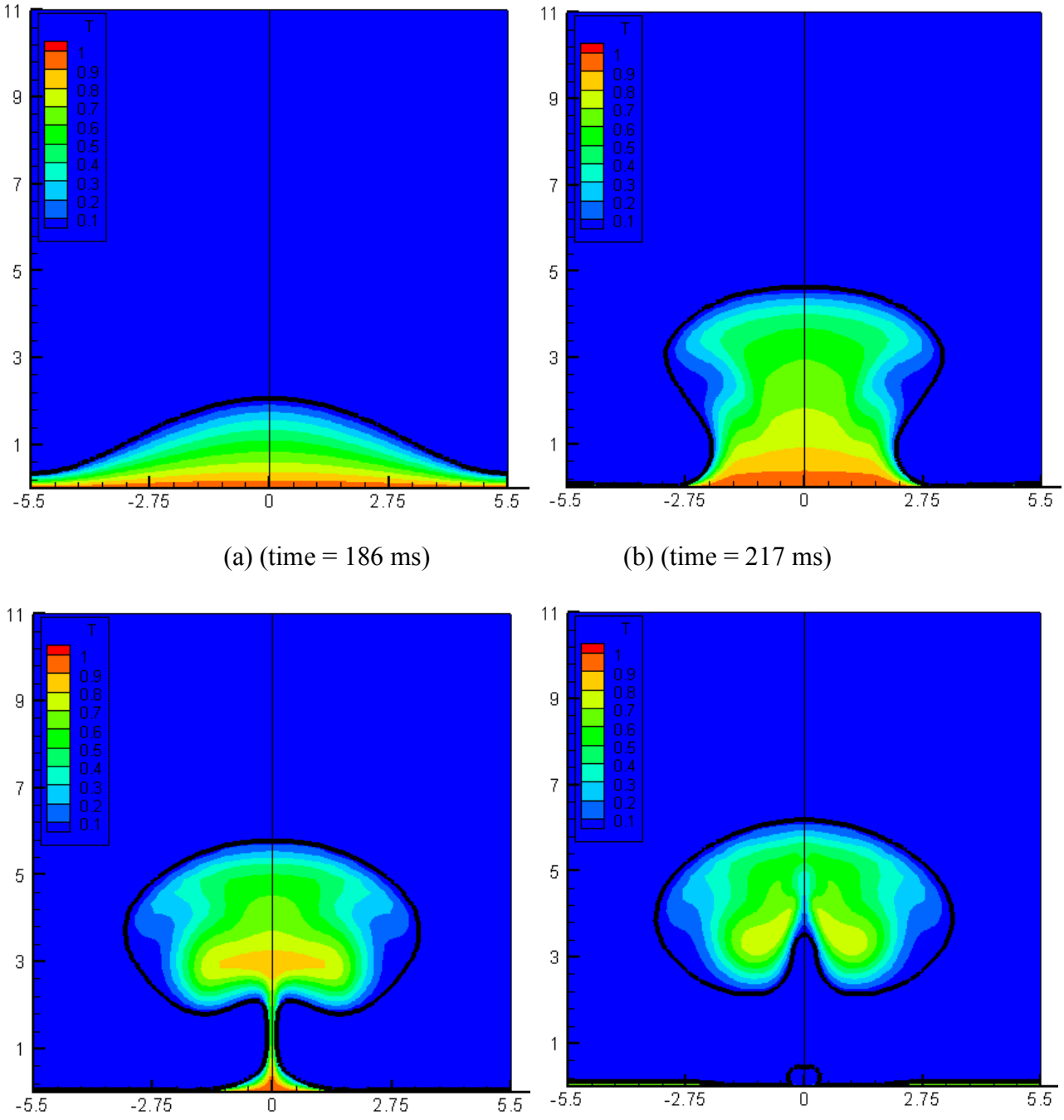
(a) (time = 0.00 ms)

(b) (time = 6 ms)

(c) (time = 16 ms)

(d) (time = 41 ms)

Figure 36 Bubble evolution at transition boiling (Z=3.49 mm, ΔT=40 °C)



(a) (time = 186 ms)

(b) (time = 217 ms)

(c) (time = 235 ms)

(d) (time = 240 ms)

Figure 37 Bubble evolution at film boiling (Z=3.49 mm, ΔT=130 °C)

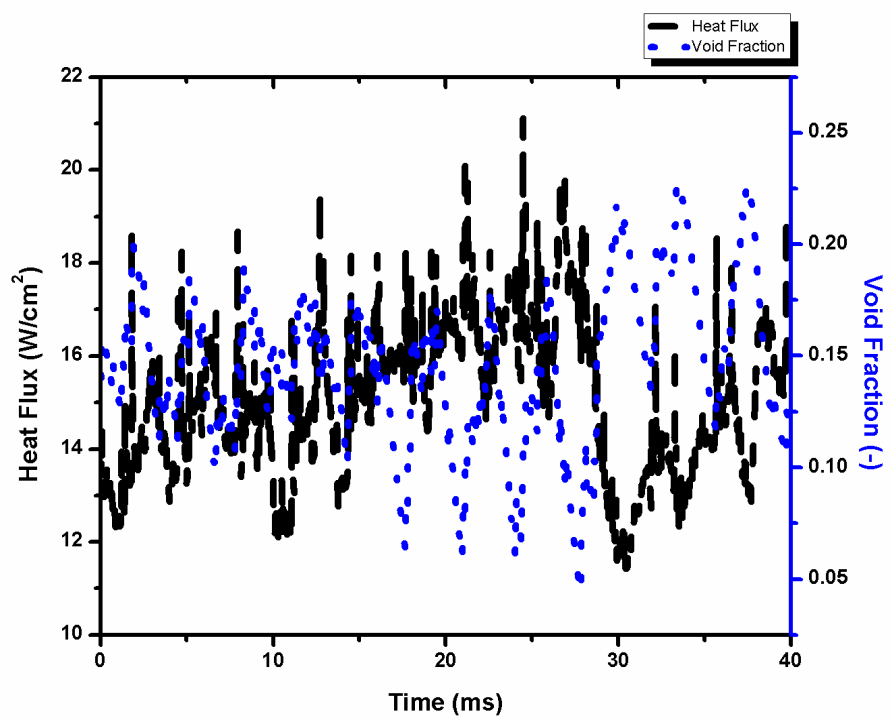


Figure 38 Variation of wall heat flux and wall void fraction (ΔT=15 °C)

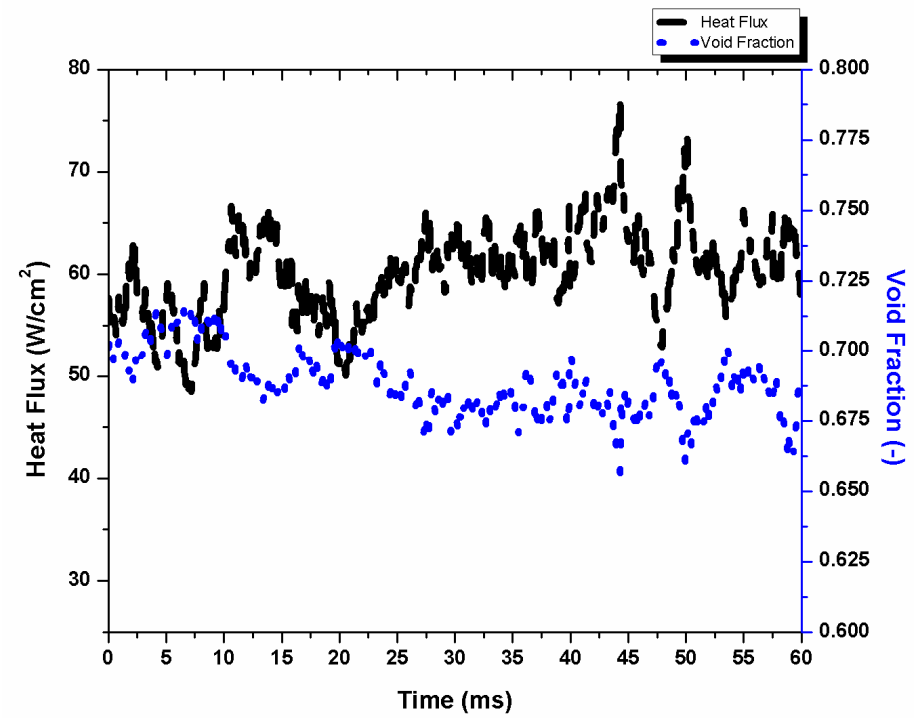


Figure 39 Variation of wall heat flux and wall void fraction (ΔT=27 °C)

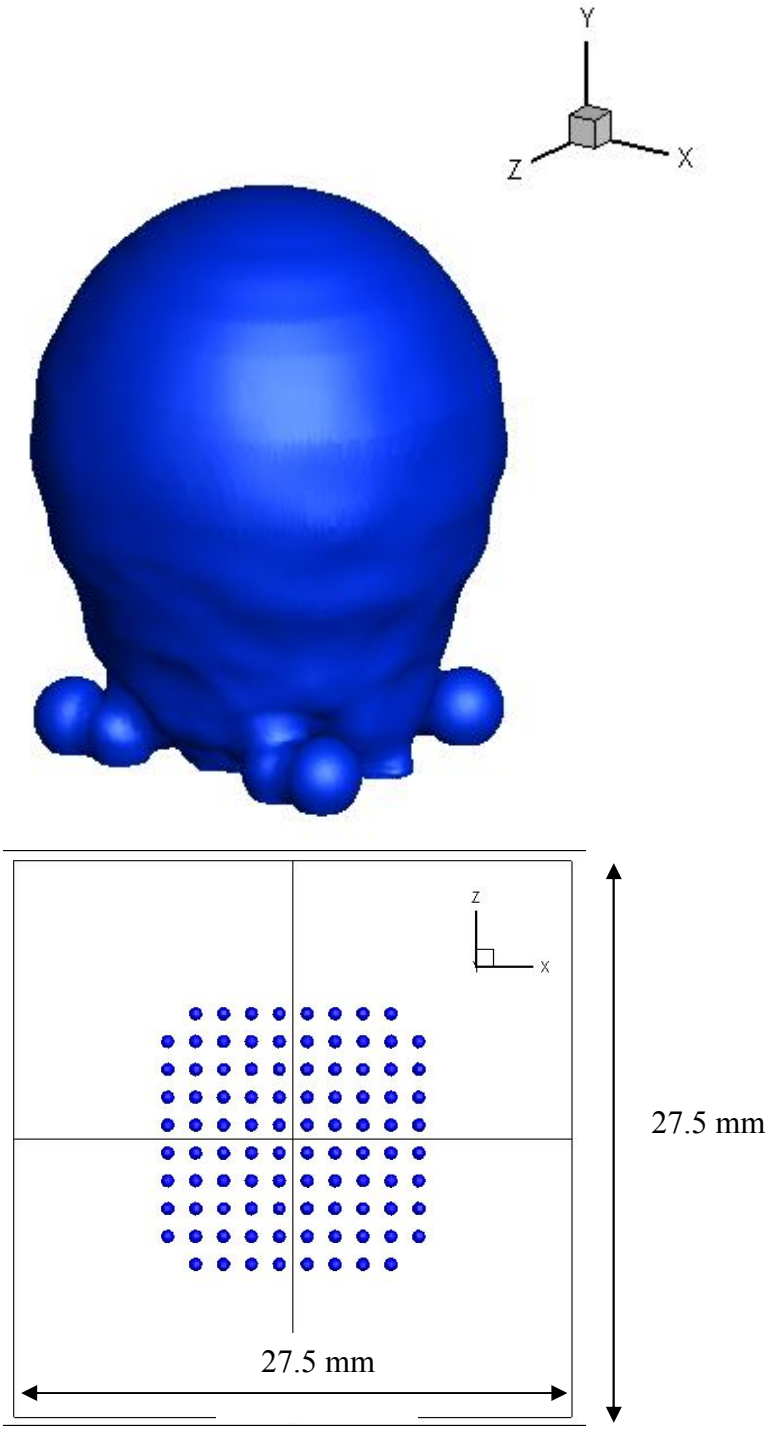


Figure 40 Growing mushroom bubble (top) along with the cavity distribution (bottom) (ΔT = 15.0 K)

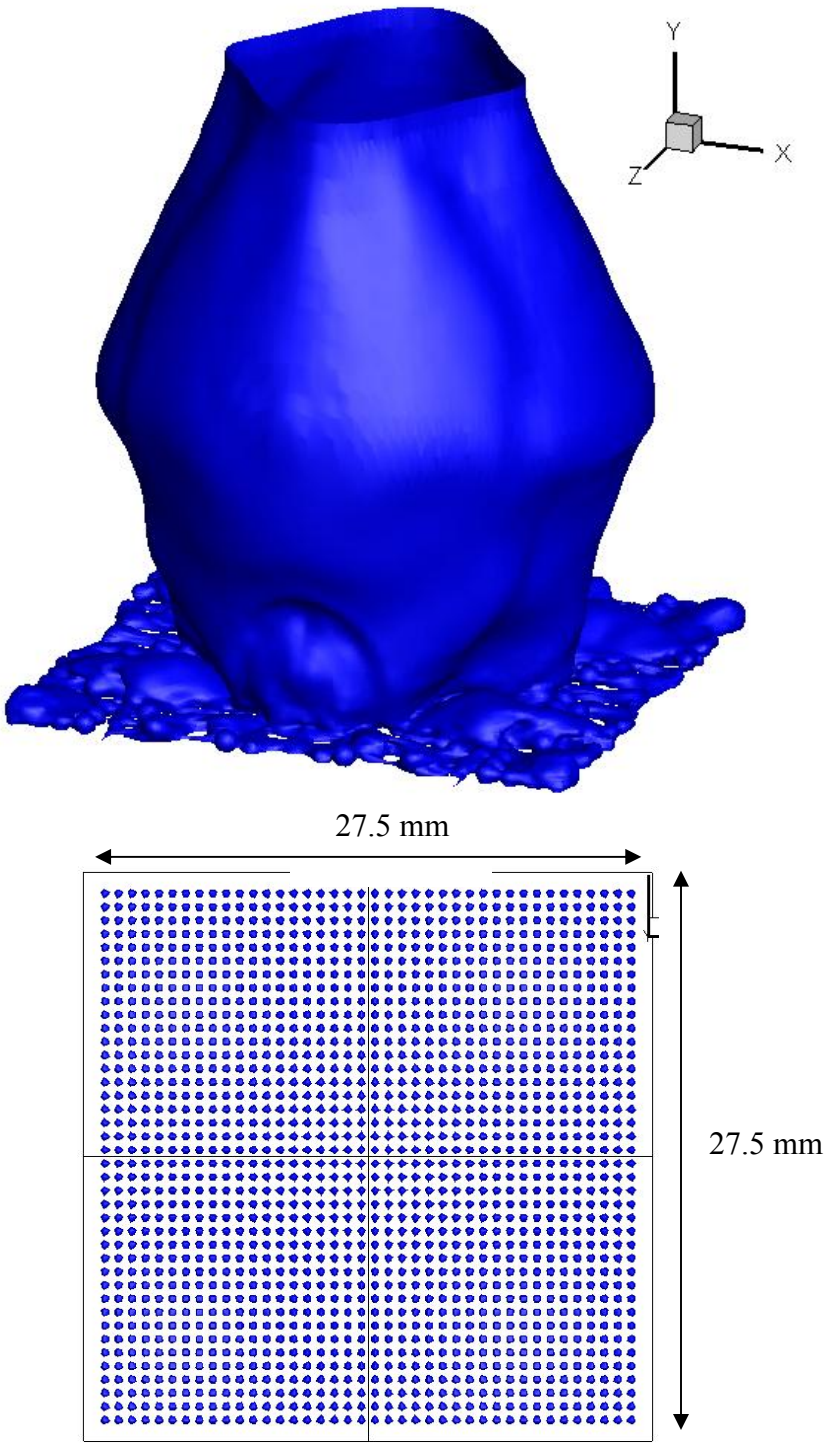


Figure 41 Formation of vapor column (top) and cavity distribution (bottom) (ΔT = 27.5 K)

Vertical Height [mm]	Vapor Fraction [-]	Jet Diameter (mm)	Vapor Velocity [m/s]	Vapor Flow Rate [mm^3/s]
6.87	0.51	11.09	0.92	89172
13.75	0.59	11.92	0.79	89126
20.62	0.45	10.49	1.03	89311
27.15	0.17	6.51	2.66	88645

Table 6 Average vapor velocity, vapor fraction and jet diameter with height at CHF

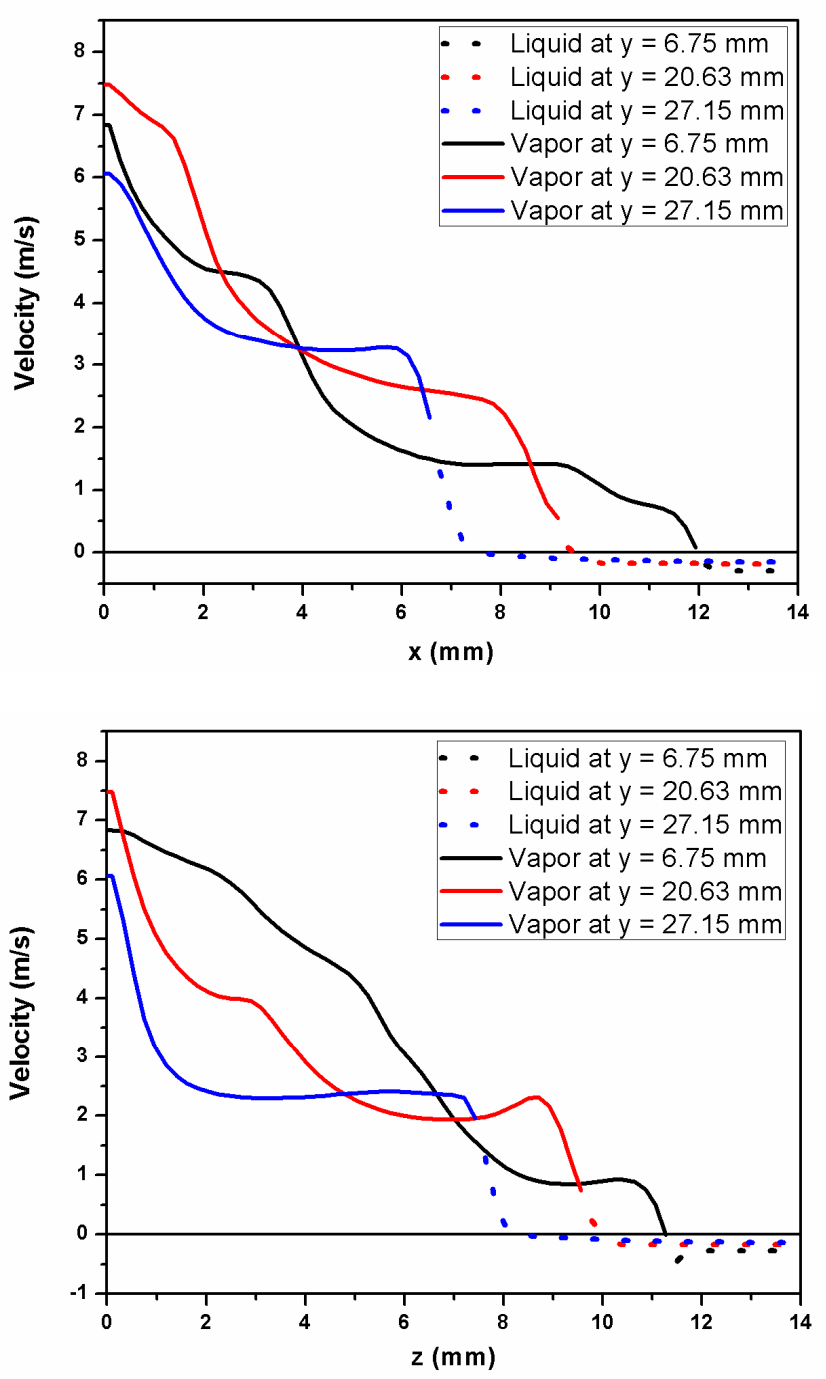


Figure 42 Vertical velocity variation with lateral distance at different heights at CHF

Figure 42 shows the planar velocity profile at three different height of the sustained vapor column seen during CHF (Figure 41). It can be seen that the centerline vapor velocity inside the vapor column has a maximum value lying in the range of about 6-8 [m/s]. Although liquid is primarily flowing down yet it has upward velocity close to the interface due to the shear forces exhibited by the outgoing vapor at the interface. However, at the vertical distance of 6.75 mm the liquid velocity has only negative values stating that liquid does not experiences significant drag. Table 6 shows the average vapor velocity, vapor fraction and equivalent jet diameter at three different height for the case of CHF.

Figure 43 shows the simulated boiling curve and comparison to the existing correlations. The nucleate boiling regime of the boiling curve is under predicted from Stephan and Abdelsalam (1980) and Rohsenow (1952) correlation with the slope of the simulated boiling curve being approximately 4. It can be seen that the differences between the Stephan and Abdelsalam (1980) correlation values and the simulated values increases from 33% approximately at the low superheat to about 50% at the higher end of nucleate boiling.

Figure 44 shows the variation of wall void fraction with superheat in comparison to the two dimensional numerical simulation performed earlier by Garg and Dhir (2015) plotted on a linear scale for clarity. Since no data for wall void fraction spanning the entire boiling curve is available for horizontal surface, comparison is made with the two-dimensional simulations performed earlier using the same numerical model. For the transition boiling regime the deviation in the values of wall void fraction is very low and it is seen that for a wall superheat of 40°C onwards more than 90% of the area is covered with vapor for both the cases.

In the transition boiling regime some comparison can still be made as experimental data is available. Figure 45 shows the variation of F-factor, fraction of liquid contact, as a function of wall superheat. Experimental values reported by Lee *et al.* (1985) are considerably different and higher than the numerical values as the F factor calculated by them is rather local liquid contact time fraction, which is the time during which the liquid makes contact with the heater surface at a particular point, rather than the fraction of liquid area on the heater surface. With the assumption that the process is ergodic (Lee *et al.* 1985), the F factor can be approximated as the ratio of liquid contact area to total area which equals to liquid contact time to total time at any given point. Data points from Ragheb and Cheng (1979) is originally for flow boiling with the assumption that F=1 at q_{max} and F=0 and q_{min} while that from Tong and Young (1974) is simply the ratio of transition boiling heat flux to critical heat flux. In the present study, the F factor is both temporal and spatial averaged and hence can be simply expressed as 1 - α (wall void fraction). As pointed out by Buchholz *et al.* (2006) due to high driving ΔT in the transition boiling, the local evaporation process is very fast and the intermittent liquid solid contacts is expected to be related to high vapor velocity. They reported that these local rewetting and vapor growth at the surface leads to temperature drop rates of upto 30 [K/ms]. In the present simulations intermittent liquid-solid contacts (frequency ~ 100 contacts s^{-1}) are seen during the simulations for the transition boiling as can be inferred from Figure 36 with 4 contacts in a time span of roughly 40 ms. For the present case some liquid solid contacts were seen in the film boiling as well with the computed F factor value being non zero at a wall superheat of 100 °C onwards till 125 °C (F-Factor of 0.0013), also it has been reported that the F factor value has a non zero value into the film boiling regime as well signifying some degree of intermittent liquid solid contacts

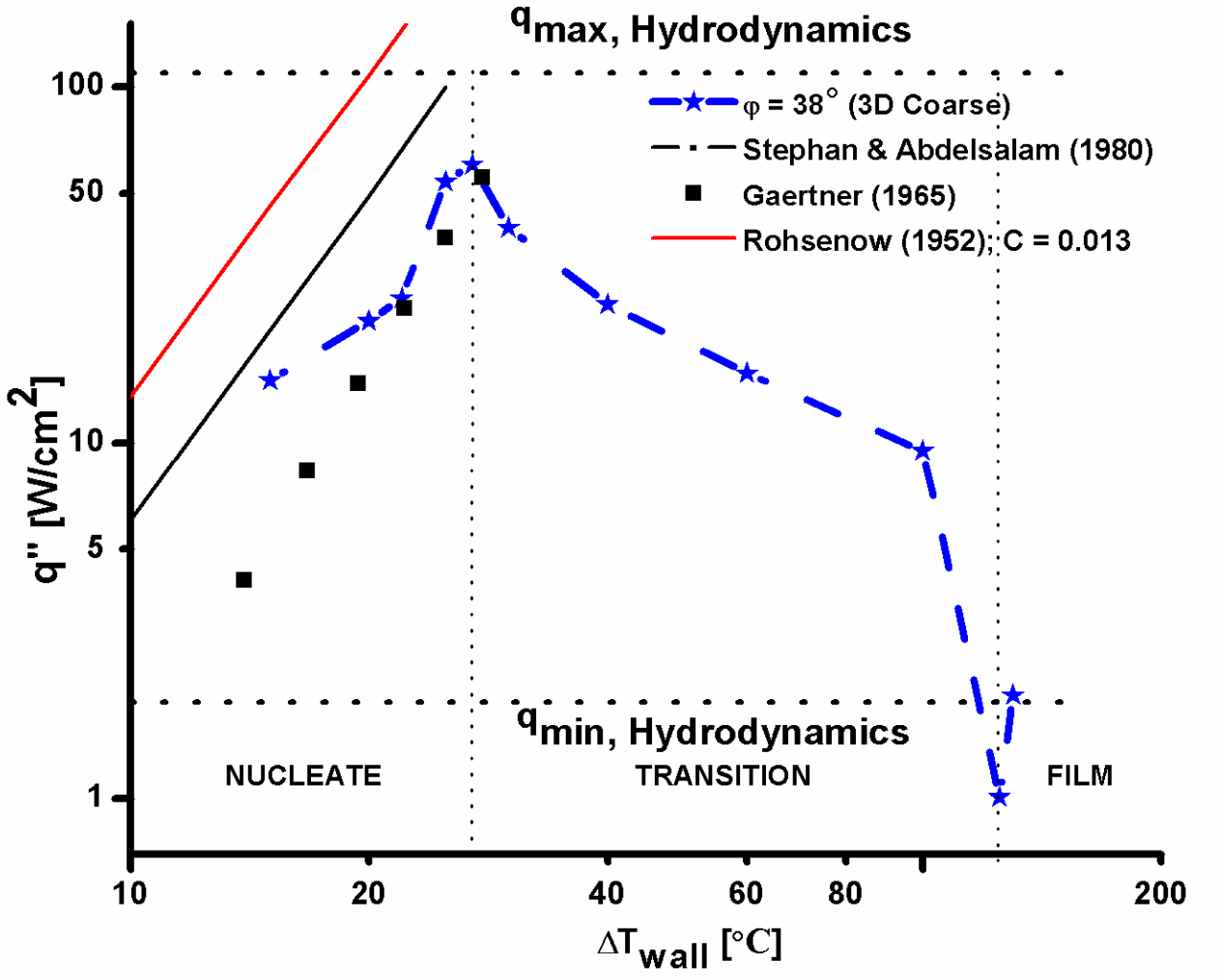


Figure 43 Simulated boiling curve

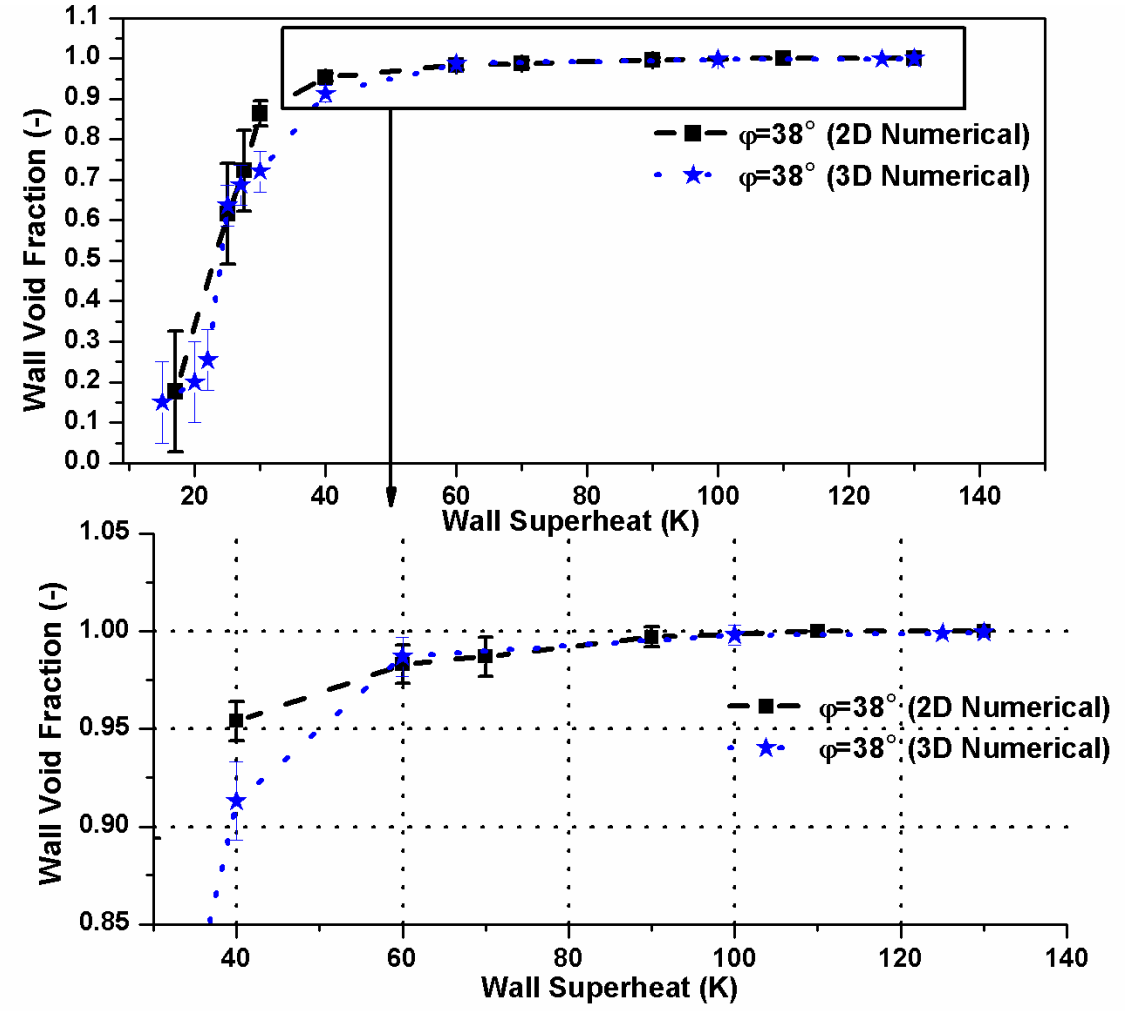


Figure 44 Variation of wall void fraction with superheat

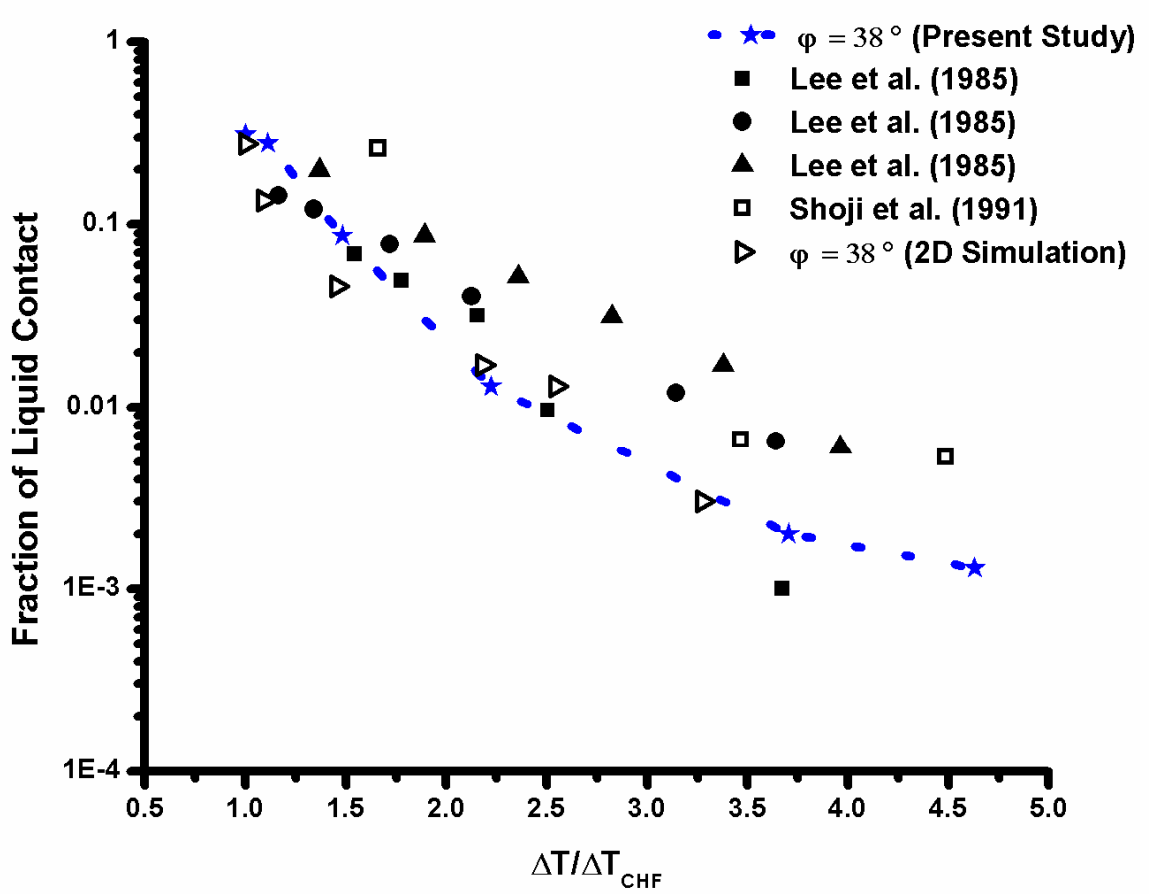


Figure 45 Variation of F factor with superheat in the transition boiling regime

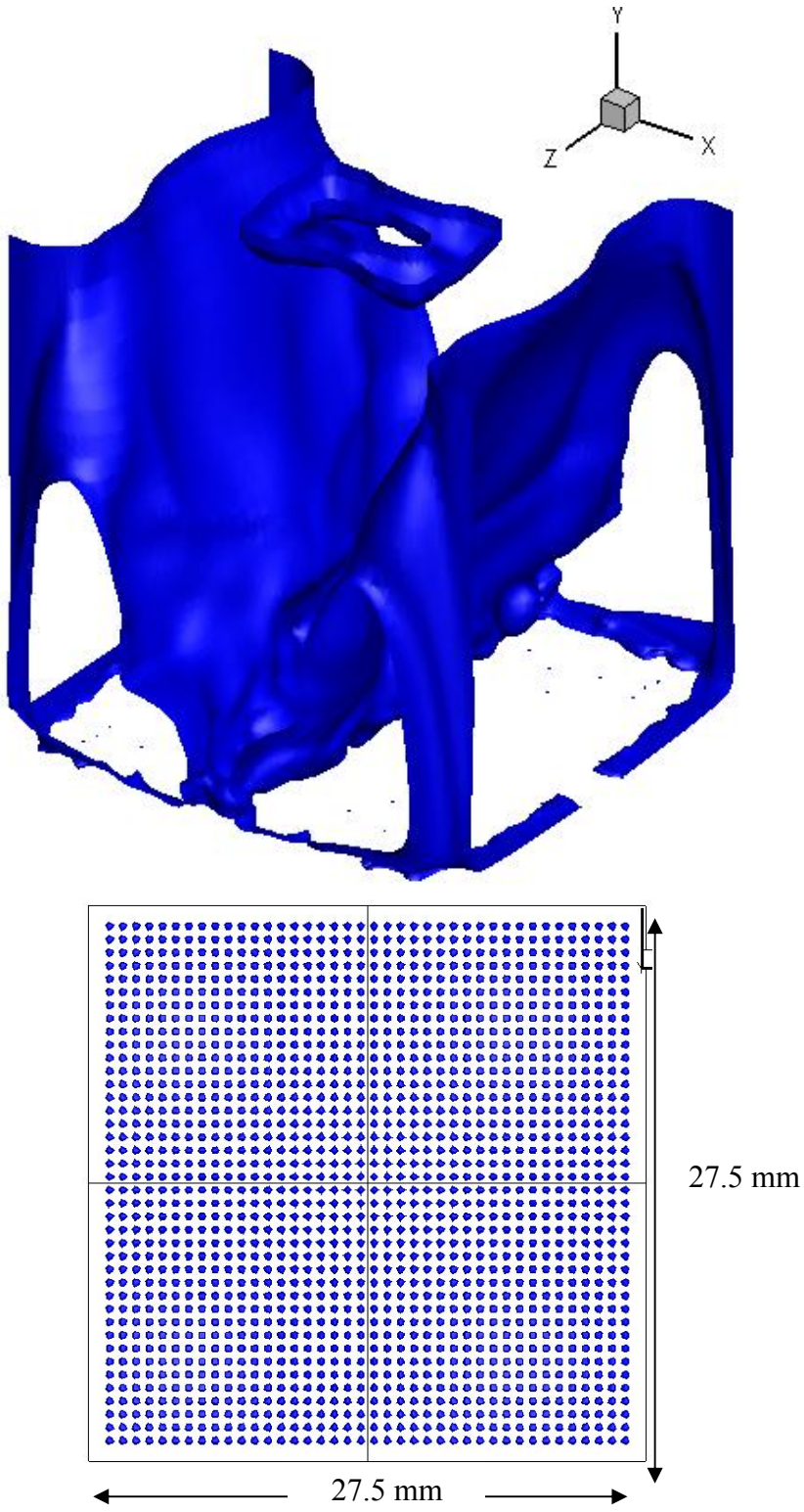


Figure 46 Vapor configuration (top) and cavity distribution (bottom) (ΔT = 40.0 K)

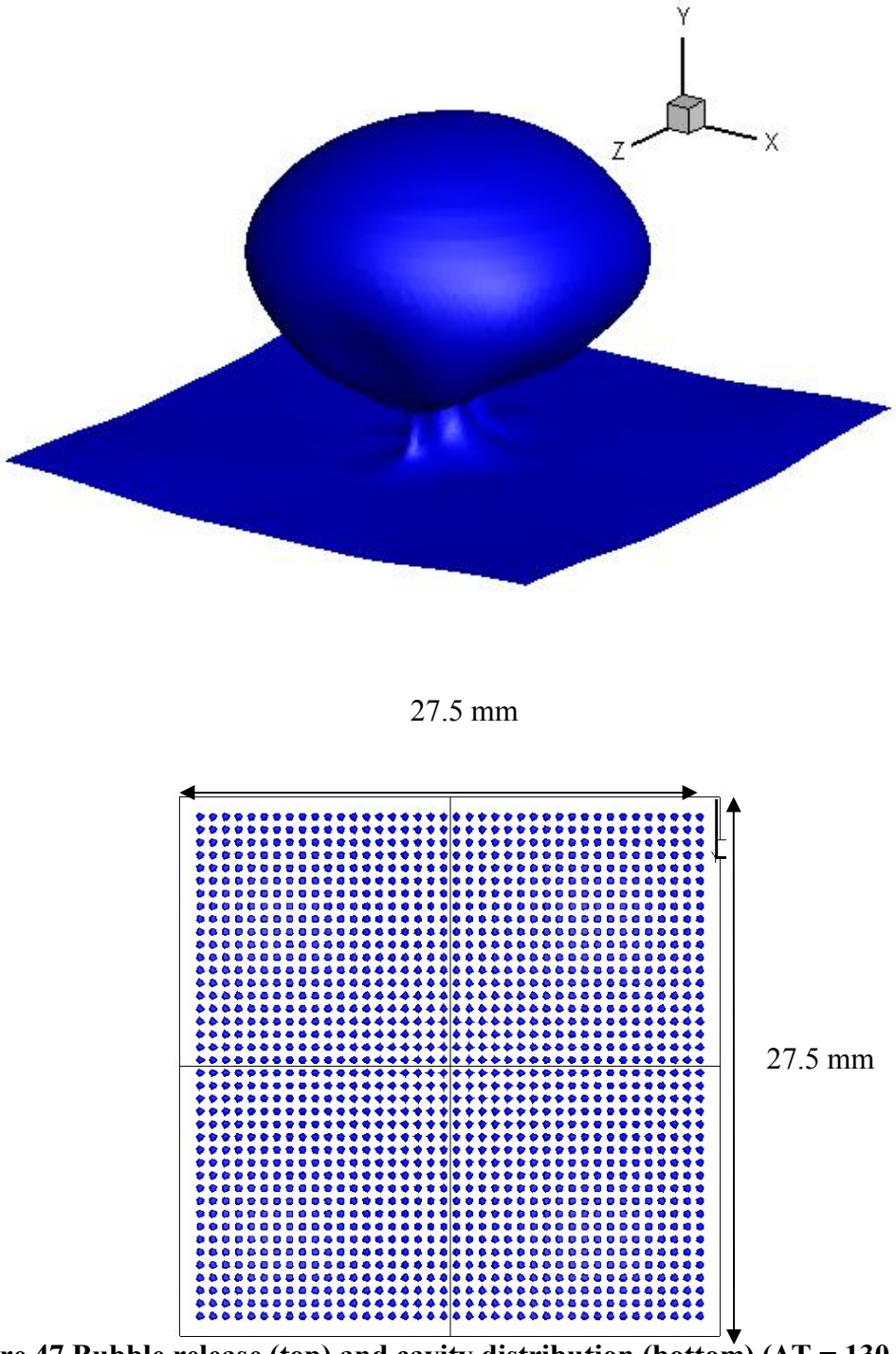
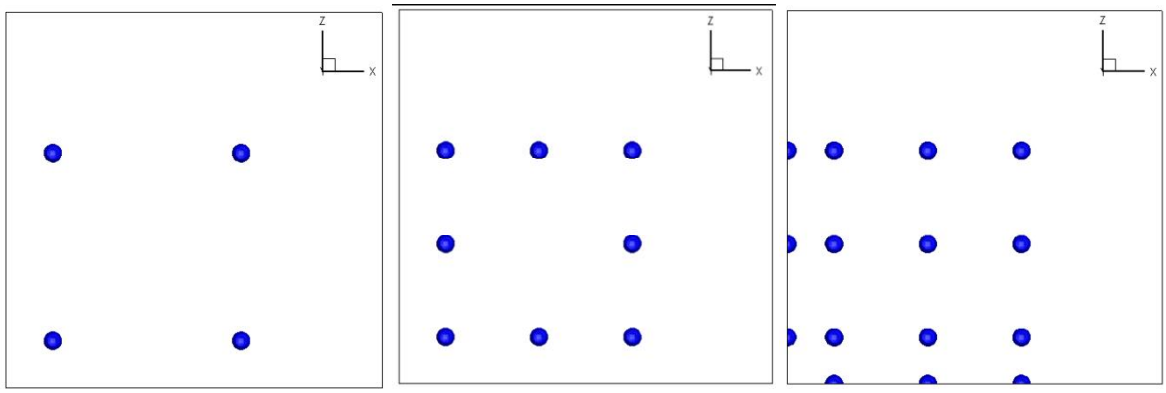


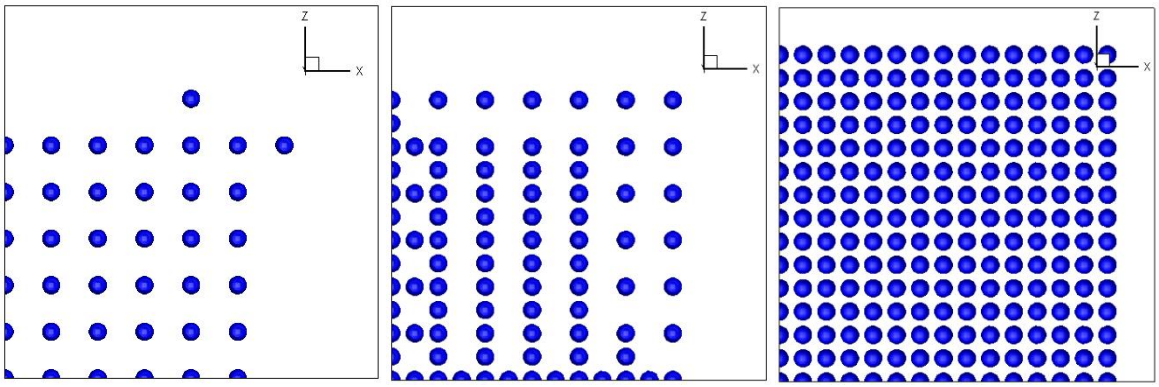
Figure 47 Bubble release (top) and cavity distribution (bottom) (ΔT = 130.0 K)

5.2 3D Fine Grid Simulations , Case II (φ=38°, φ=69°)

The case described in the present section case was run for finer grid with the intention of resolving the anomaly observed in the trend for nucleate boiling regime for the two-dimensional cases. Simulation were run for two contact angles of 38° and 69° with the quarter domain extents being $0 \leq x \leq 2$, $0 \leq y \leq 10$ and $0 \leq z \leq 2$ (non dimensionalized form) resolved with a grid density of 98 x 194 x 98. The cavity distributions for both the contact angles are shown in Figure 48 and Figure 49 while the simulated boiling processes are shown in Figure 50 - Figure 53. It has to be noted here that the cavity points are marked by circles as shown in Figure 48 and Figure 49 with the cavities in close proximity (but within) to the boundaries marked by semi-circles. The simulated nucleate boiling curves are compared against coarse grid results as shown in Figure 54. Unlike the two-dimensional case the nucleate boiling heat flux is higher for larger contact angles for the three-dimensional finer grid case. The contact angle of 69° has higher nucleate boiling heat flux than 38° case as the nucleation site density is more for the same superheat. However the maximum heat flux for the 38° is more than 69° case with the CHF values being 60.4 [W/cm^2] and 54.8 [W/cm^2] for 38° and 69° respectively. Figure 55 shows the variation of wall void fraction computed for the fine grid cases in comparison to the coarse grid.

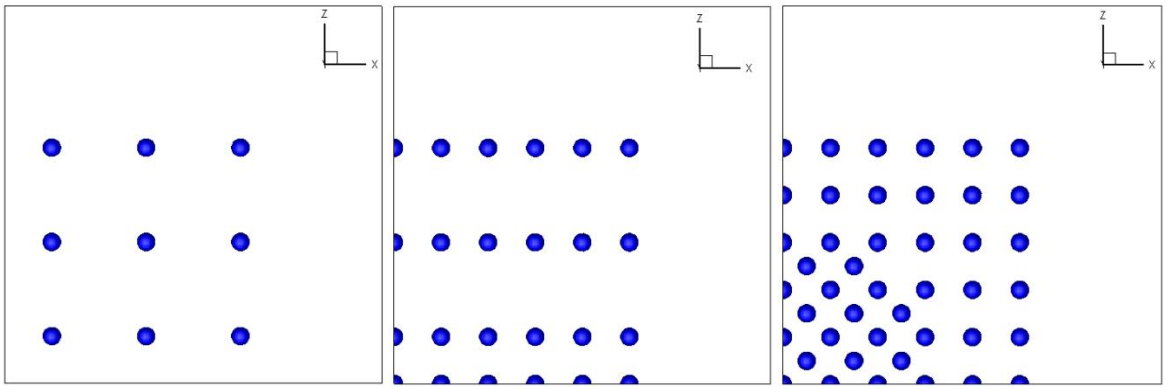


ΔT = 15 [K] ΔT = 18 [K] ΔT = 20 [K]

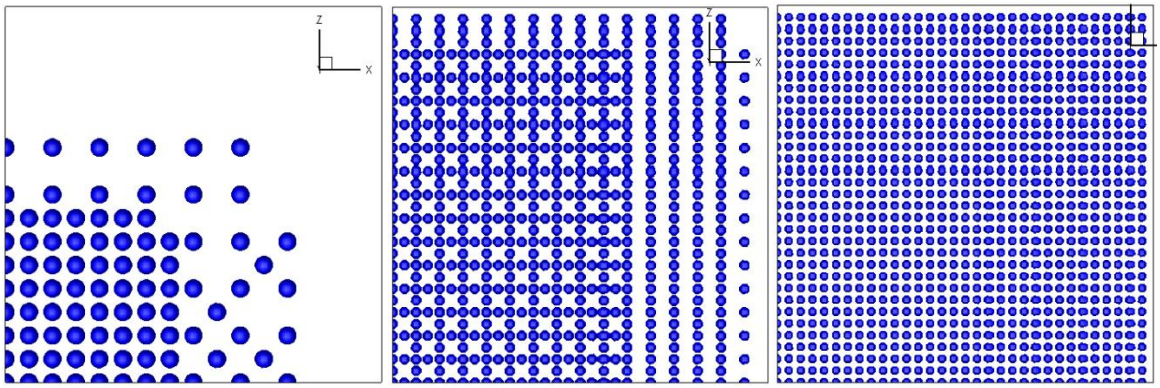


ΔT = 24 [K] ΔT = 28 [K] ΔT = 33.5 [K]

Figure 48 Nucleation sites distribution for contact angle of 38°.
(Field of view 5 mm x 5 mm)



ΔT = 15 [K] ΔT = 18 [K] ΔT = 20 [K]



ΔT = 23 [K] ΔT = 33.5 [K] ΔT = 40.0 [K]

Figure 49 Nucleation sites distribution for contact angle of 69°.
(Field of view 5 mm x 5 mm)

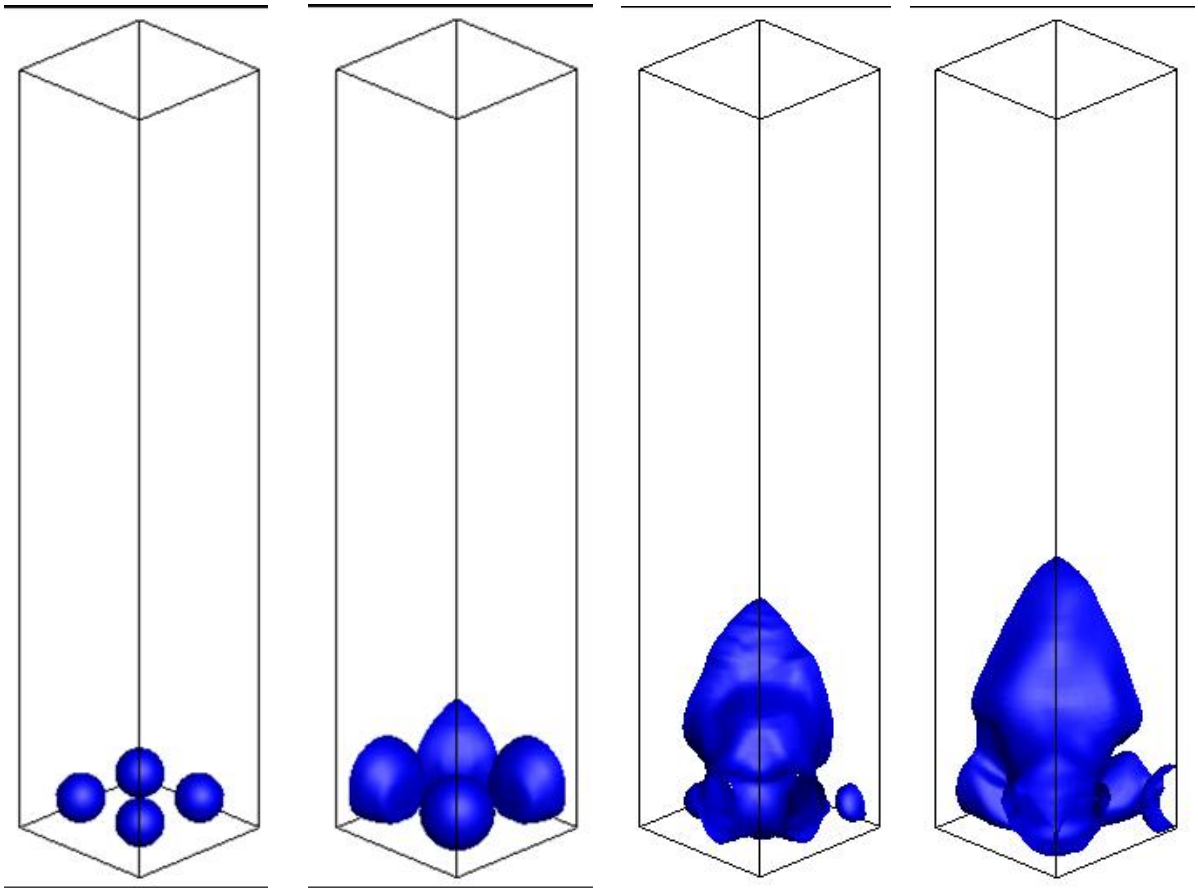


Figure 50 Boiling process for φ = 38°, ΔT = 15 [K]
(Field of view 5 mm x 25 mm x 5 mm)

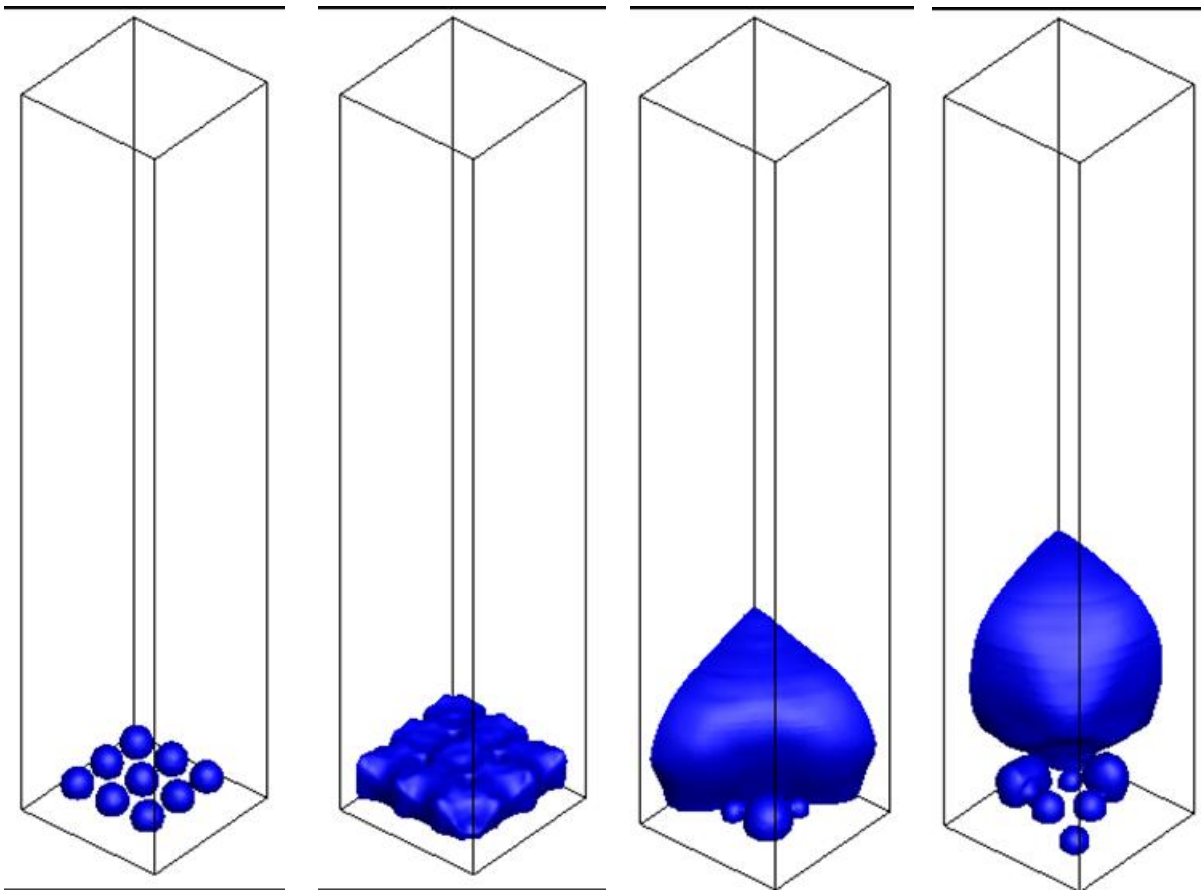
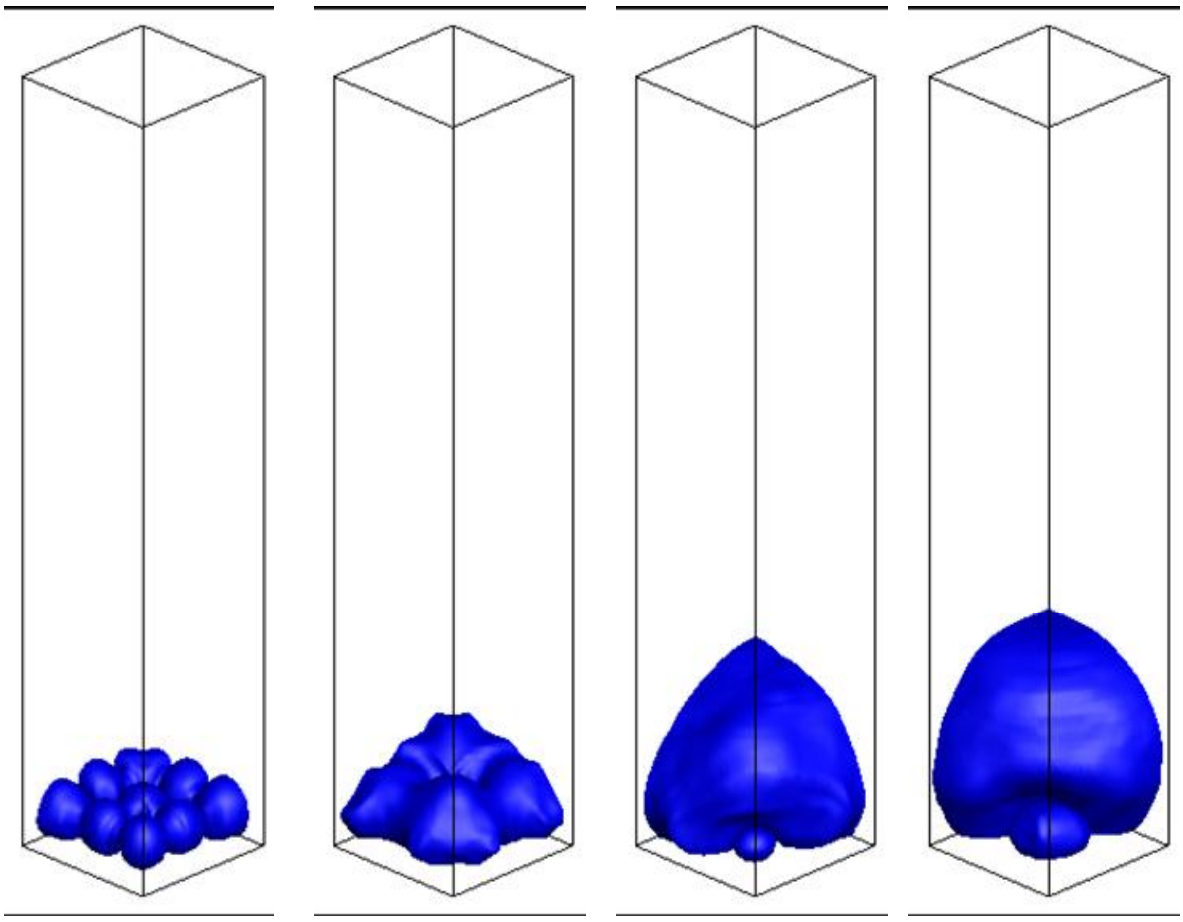


Figure 51 Boiling process for φ = 69°, ΔT = 15 [K]
(Field of view 5 mm x 25 mm x 5 mm)



Time = 2.52 ms | Time = 5.46 ms | Time = 13.6 ms | Time = 21.4 ms

Figure 52 Boiling process for φ = 38°, ΔT = 20 [K]
(Field of view 5 mm x 25 mm x 5 mm)

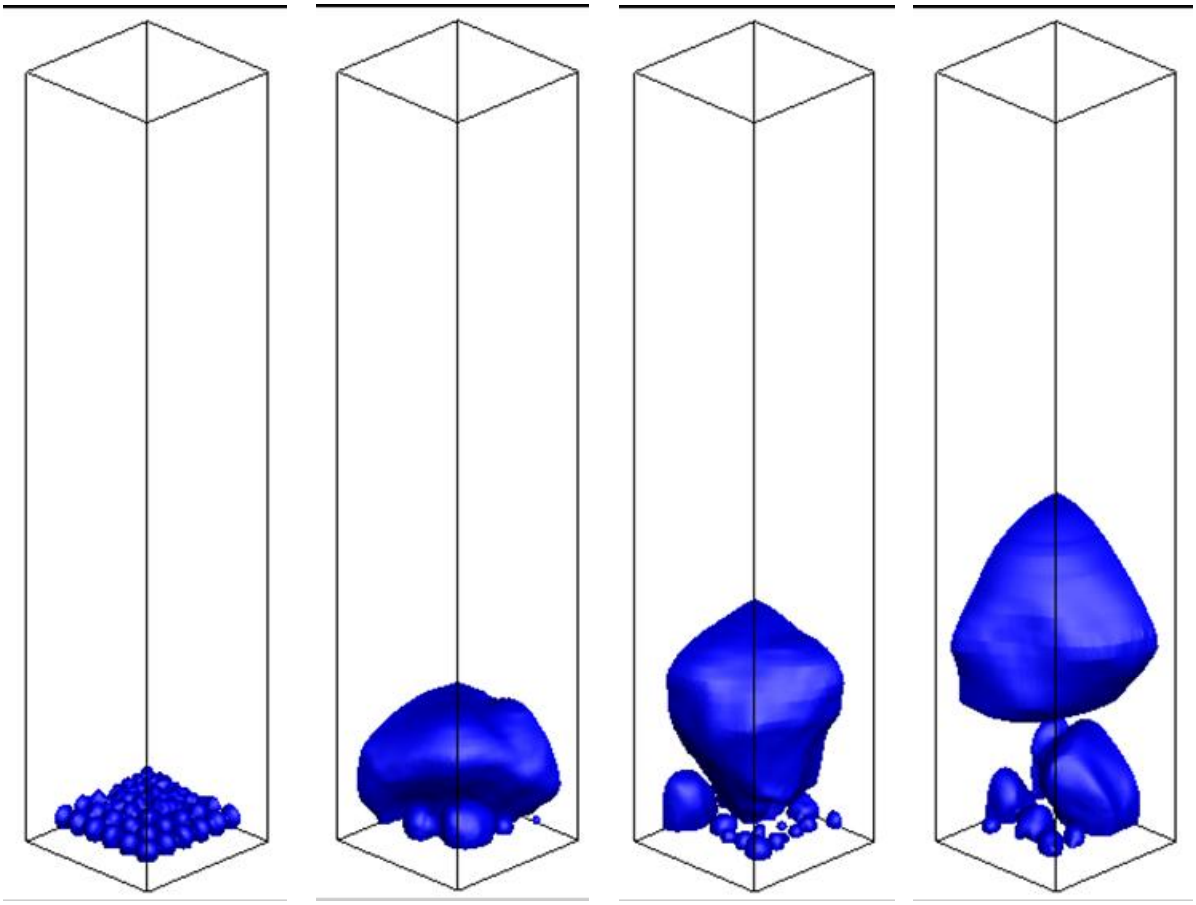


Figure 53 Boiling process for φ = 69°, ΔT = 20 [K]
(Field of view 5 mm x 25 mm x 5 mm)

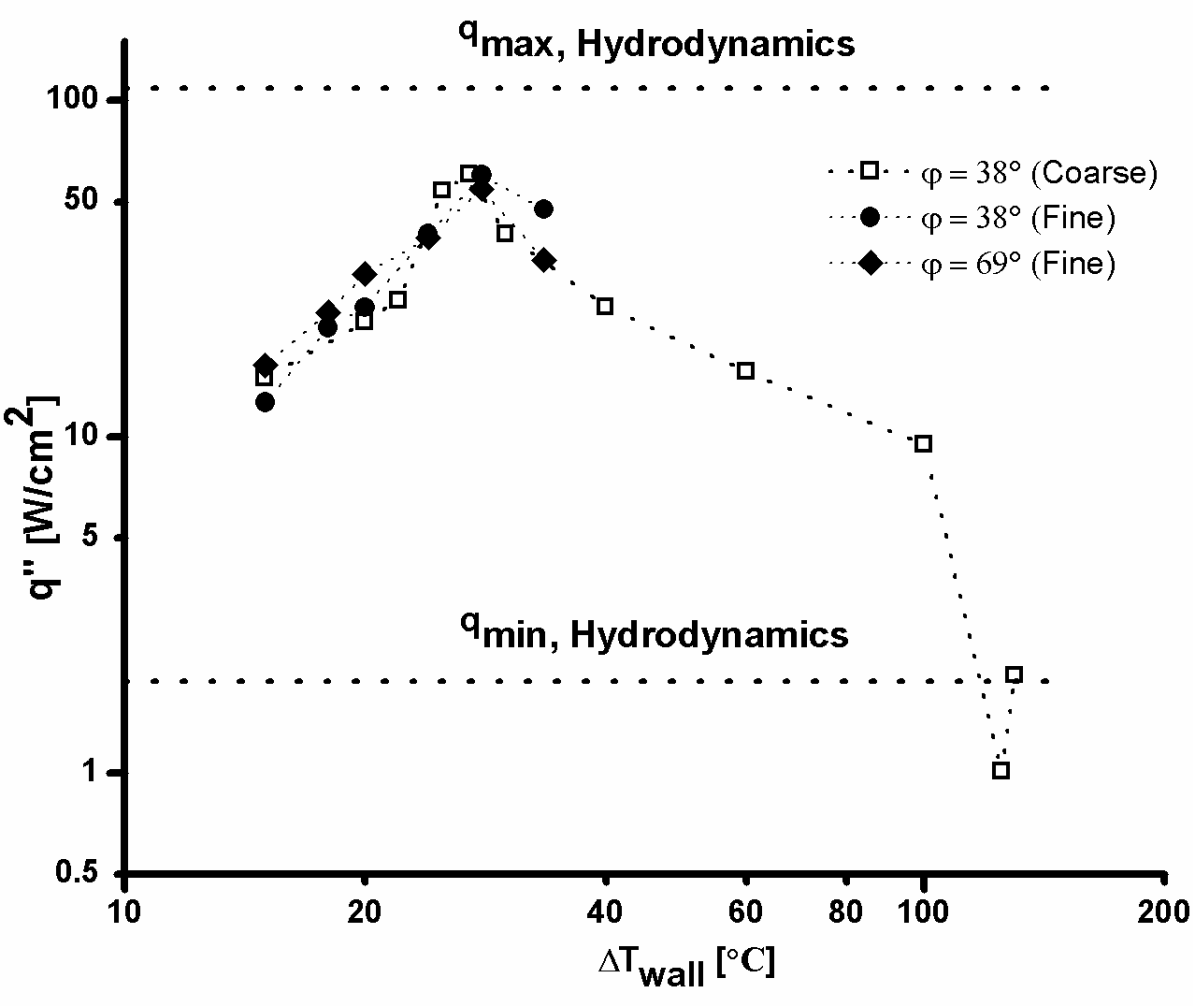


Figure 54 Comparison of boiling curves for 3D cases

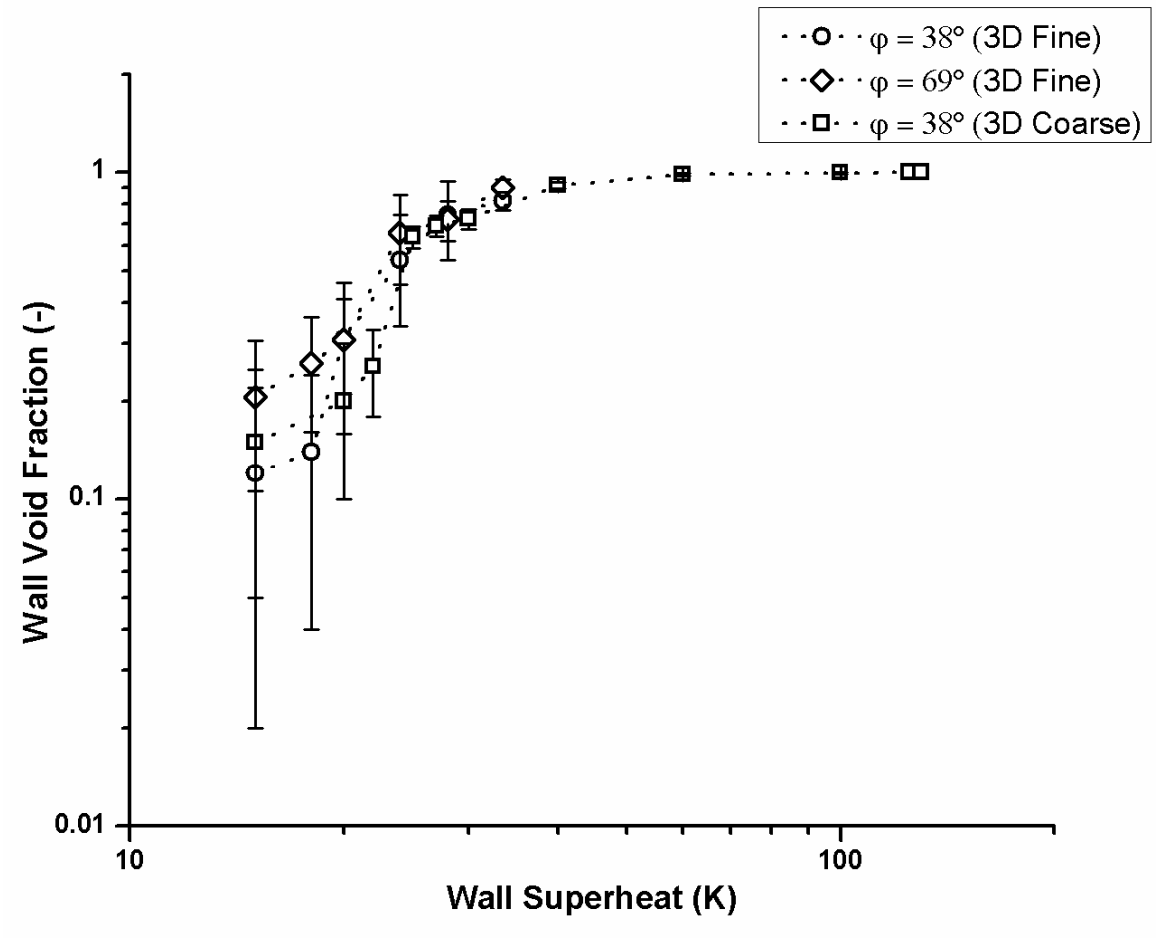


Figure 55 Comparison of wall void fraction for 3D cases

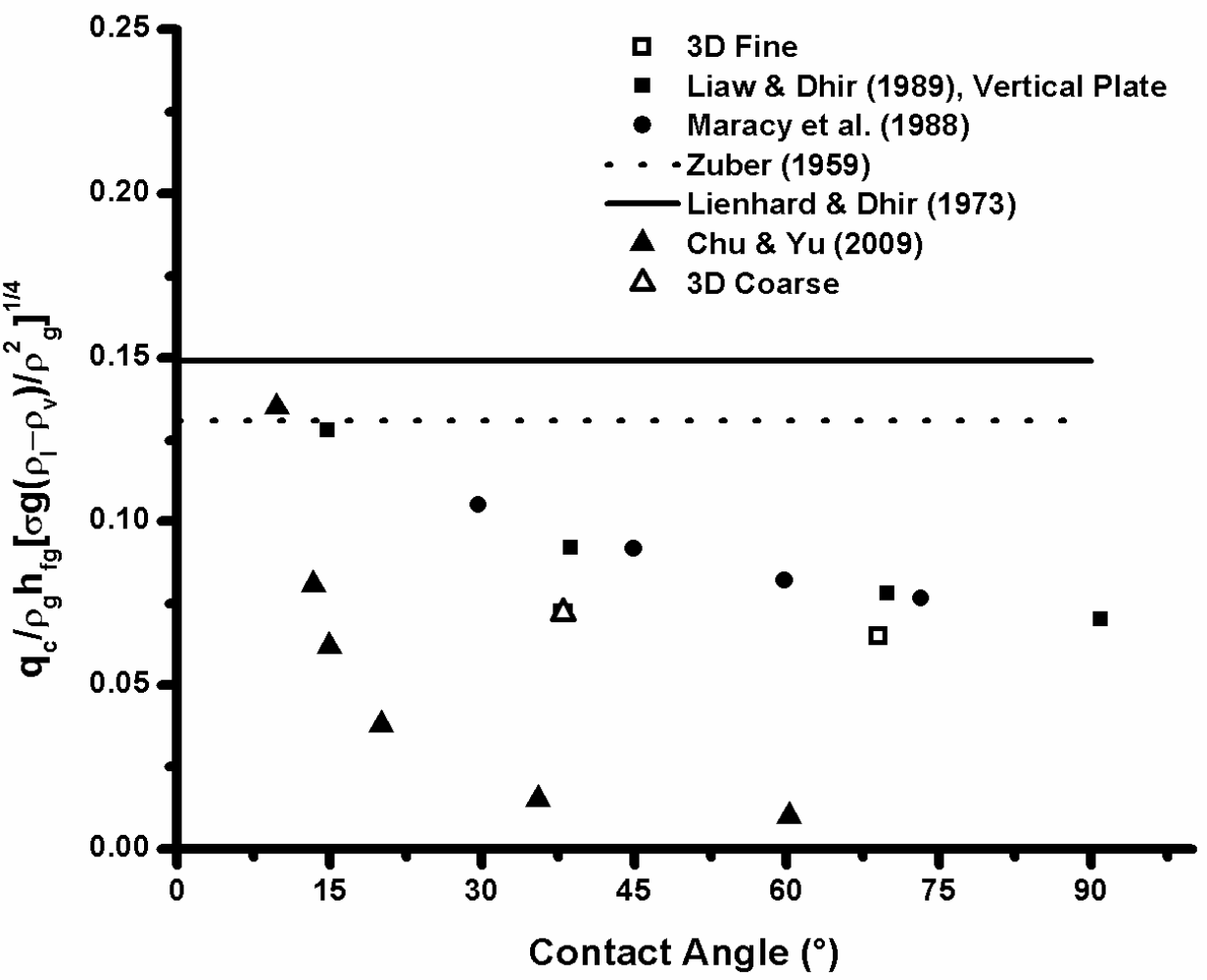


Figure 56 Variation of CHF with contact angle

Figure 56 shows the comparison of computed CHF, which is under predicted for both the contact angles in the present study, against value previously reported by Maracy and Winterton (1988) and Liaw & Dhir (1989). Maracy and Winterton (1988) computed the boiling heat flux using Kudryavtsev two point method which requires two known temperature values as a function of time.

Figure 57 - Figure 59 shows the energy partitioning from the wall into superheating the liquid, interface and microlayer for the three-dimensional cases. The comparison of the energy partitioning has been done with respect to both wall void fraction and wall superheat . The fine grid cases haven`t been run long enough because of which the trend seen is inconclusive. A rough estimate for the duration for which fine grid cases should be run is approximately $t^* > 5$. The energy going into superheating the liquid is calculated by subtracting the energy utilized for vapor production from the energy provided by the wall. The coarse grid ($\varphi = 38°$) yields a downward trend with wall void fraction for the fraction of energy going into superheating the liquid in comparison to the finer grid counterpart. The fraction of energy going into superheating of liquid is around 85% for the coarse grid case while it is around 50% for the fine grid cases at 15°C. Previously, Dhir (2006) had reported 70% energy from the wall going into superheating the liquid at a wall superheat ($\varphi = 54°$, $\Delta T = 8$ °C). Finally, the fraction of energy utilized for superheating the liquid drops to almost zero for a wall superheat of 130°C as all of the wall heat flux goes into vapor.

Figure 58 shows the energy fraction going through the interface (excluding the microlayer) which for the coarse grid case ($\varphi = 38°$) is around 23% at CHF, has a value of 65% for most of the transition boiling regime and ultimately attains a value of 100% at a wall superheat of 130 °C due

to entire surface being covered with vapor film. The energy fraction going into the microlayer is shown in Figure 59 which for the coarse grid ($\varphi = 38°$) peaks at the critical heat flux case with the value of around 45% and finally goes to zero at wall superheat of 130 °C. Experiments done by Stephan *et al.* (2009) also concluded that 50-60% of the energy fueling bubble growth comes through microlayer. Since, Microlayer contribution increases linearly with superheat and depends on contact line density, for coarse grid case a near about constant value of 25% -30% is observed even as the wall superheat increases from 40 °C to 100 °C as the dry area is increased from 90% to 99% over the same range. Fine grid cases for both the contact angles show approximate value of 40% of the energy being transferred into vapor through the interface for the range of wall superheat considered. For contact angle of 38° about 40% of the energy has been computed for superheating the liquid while the 69° counterpart shows a slightly higher value of 50% for the same range of superheats. Microlayer contribution for both the contact angles has been found to be much lower in comparison to the coarse grid case with an approximate value of 10-20 % with 38° contact angle being approximately 5% more than 69° over the range of wall void fraction plotted as shown in Figure 59.

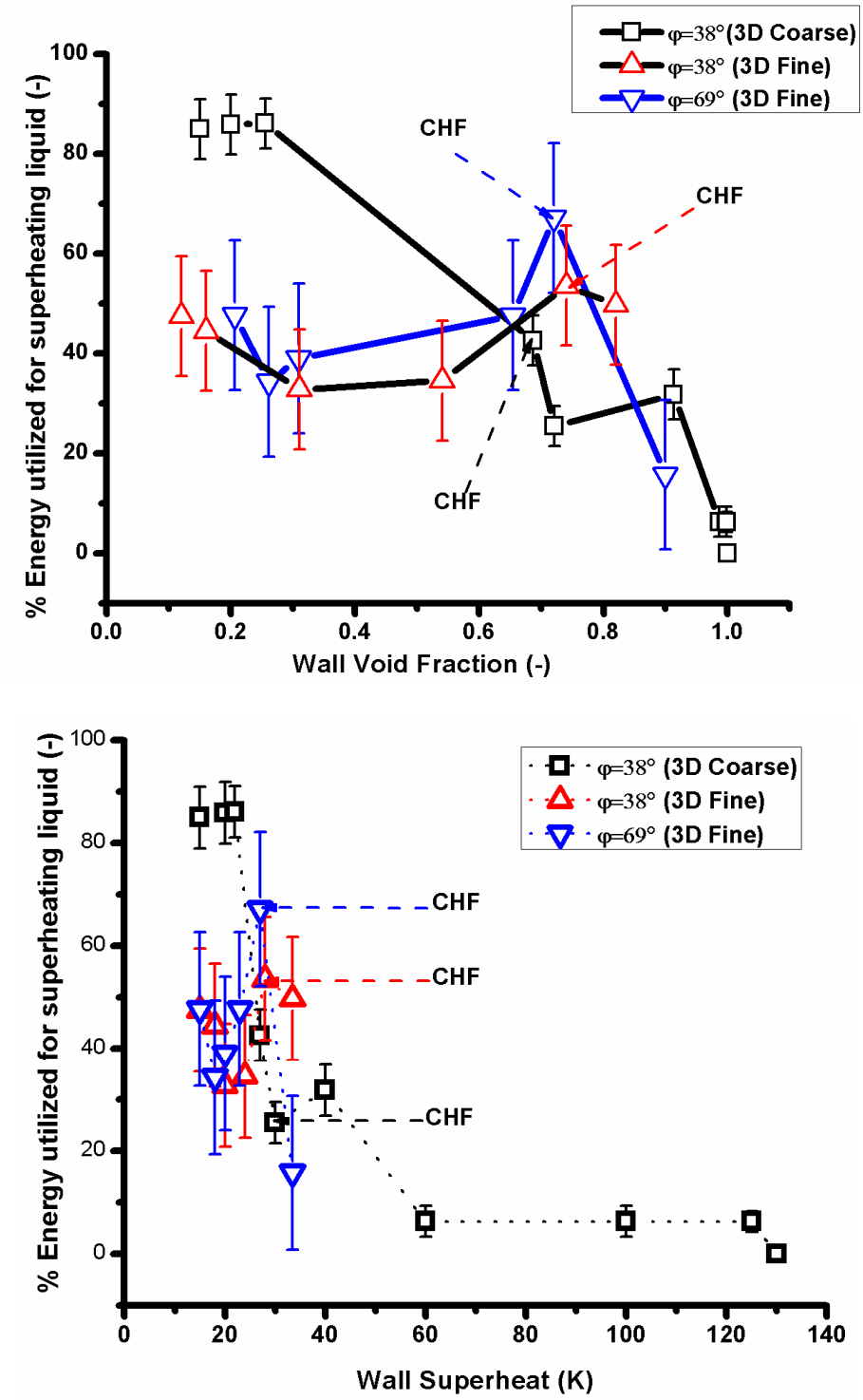


Figure 57 Energy fraction utilized for superheating of liquid for 3D cases

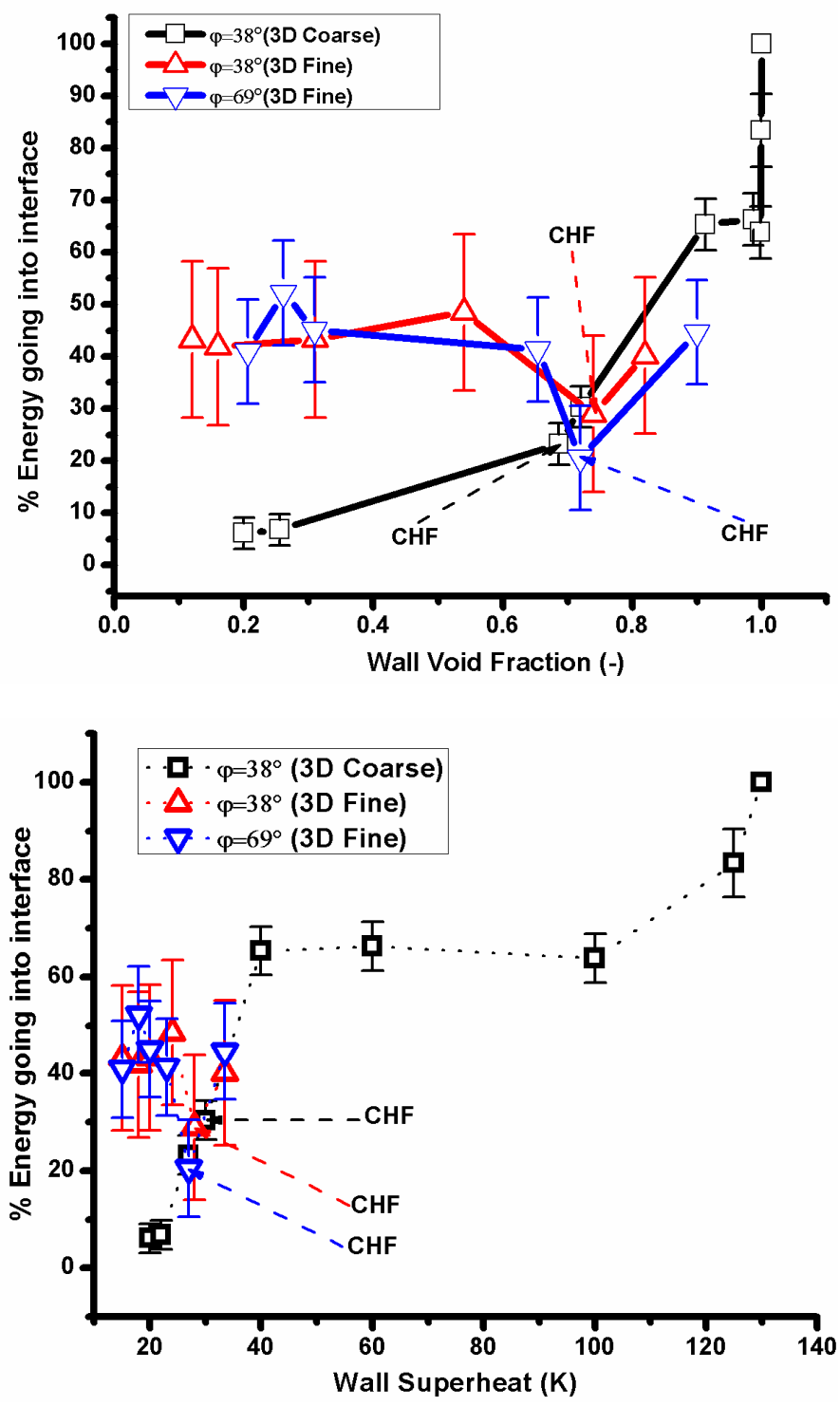


Figure 58 Energy fraction going into interface for 3D cases

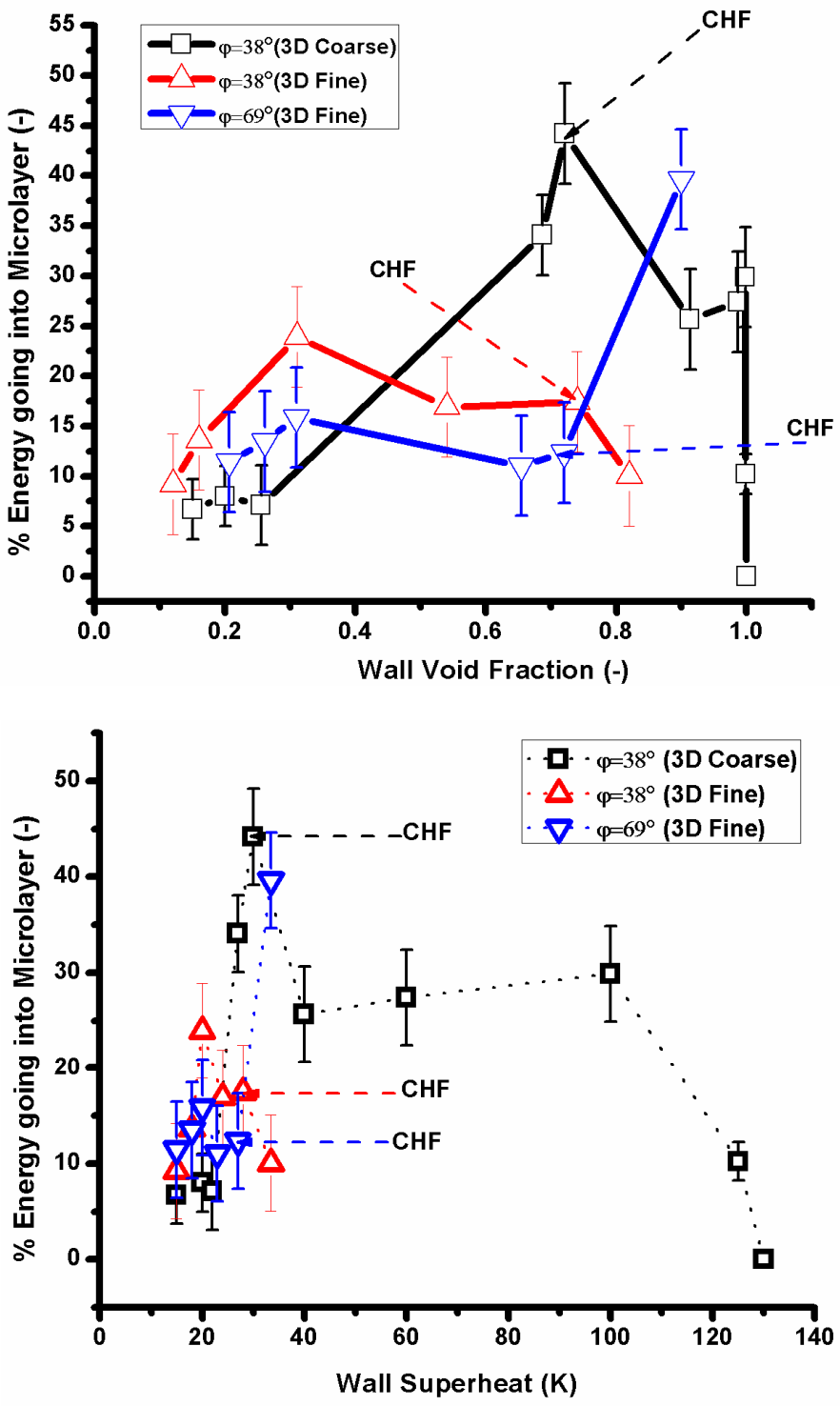


Figure 59 Energy fraction going into microlayer for 3D cases

5.4 Conclusion

This chapter focused on three-dimensional simulation of the entire boiling curve as two-dimensional approximation is inadequate in capturing the essential physics of the process. The trend observed for the nucleate boiling heat flux regime with variation in contact angles is correct as seen in the fine grid case and is different than the two-dimensional studies done earlier. For the 3D coarse grid case at CHF, sustained superheated vapor column is seen with the CHF value computed to be the same with that obtained from fine grid. For fine grid cases the CHF value decreased with increase in contact angle. The average vapor velocity at CHF condition for 3D coarse grid was found to be approximately 1 [m/s] which is only 25% of the theoretical critical Helmholtz vapor velocity. Intermittent liquid-surface contacts are seen for the transition boiling case. Upon increasing the superheat further, the entire surface gets covered with a superheated vapor film thus entering the film boiling regime.

Summary

Both two-dimensional and three-dimensional numerical simulations of the entire boiling curve in a temperature controlled mode have been performed by a unified numerical model under the framework of parallel computing in conjunction with correlations specifying nucleation site density and bubble waiting time as functions of wall superheat and contact angle.

- The two-dimensional assumption renders the numerical model incapable of correctly predicting the trend of nucleate boiling heat flux with variation in contact angles due to the inbuilt manifestation of symmetric boundary conditions along the frontal plane of the domain. The trend of CHF with variation in contact angle is however correct.
- The three-dimensional simulations (coarse grid cases) show the rising mushroom shaped vapor bubbles in the nucleate boiling regime. Sustained vapor column with average vapor velocity of approximately 1 [m/s] is seen for maximum heat flux conditions. The transition boiling shows intermittent liquid surface contacts and upon increasing the superheat eventually the entire surface gets covered with vapor film characterized by film boiling regime. Energy partitioning from wall into liquid shows decreasing trend with superheat.
- Three-dimensional simulations with fine grids captures the correct trend in the nucleate boiling heat flux (till CHF) with contact angle

References

1. Abarajith, H.S. , Dhir, V.K., Son, G., "Numerical Simulation of the Dynamics of Multiple Bubble Merger during Pool Boiling under Reduced Gravity Conditions", *Multiphase Science and Technology*, Vol. 18, No. 1, pp. 27-54, 2006.

2. Aslam, T.D., "A Partial Differential Equation Approach to Multidimensional Extrapolation", *J. Comp. Phy,* Vol. 193, pp. 349-355, 2003.

3. Auracher, H., Marquardt, W., "Heat Transfer Characteristics and Mechanisms along the Entire Boiling Curves under Steady-State and Transient Conditions", *International Journal of Heat and Fluid Flow*, Vol. 25, pp. 223-242, 2004.

4. Basu, N., Warrier, G., R., Dhir, V., K., "Onset of Nucleate Boiling and Active Nucleation Site Density During Subcooled Flow Boiling", *Journal of Heat Transfer,* Vol. 124, pp. 717-728, 2002.

5. Basu, N., Warrier, G., R., Dhir, V., K., "Wall Heat Flux Partitioning during Subcooled Flow Boiling: Part 1-Model Development", *Journal of Heat Transfer,* Vol. 124, pp. 717-728, 2005.

6. Berenson, P.J., "Film Boiling Heat Transfer from a Horizontal Surface", *J. Heat Transfer,* 83(3), pp. 351-356, 1961.

7. Bjonard, T.A, Griffith, P, PWR blowdoen heat transfer. Symp. on the Thermal and Hydraulic aspects of Nuclear Reactor Safety. The ASME Winter Annual Meeting, 1977.

8. Brackbill, J.U., Kothe, D.B., and Zemach, C., "A Continuum Method for Modeling Surface Tension", *J. Comp. Phy,* Vol. 100, pp. 335-354, 1992.

9. Buchholz, M., Auracher, H., Luttich, T., Marquardt, W., “A Study of Local Heat Transfer Mechanisms along the entire Boiling Curve by means of Microsensors”, *Int. Journal of Thermal Sciences,* Vol. 45, pp. 269-283, 2006.

10. Carey, V., P., “Liquid-Vapor Phase Change Phenomena”, *2nd ed, Taylor and Francis*, 2007.

11. Chekanov, V.V., "Interaction of Centers in Nucleate Boiling", *Teplofyz. Vys. Temp,* Vol. 15, pp. 121-128, 1977.

12. Chow, E., “A Priori Sparsity Patterns for Parallel Sparse Approximate Inverse Preconditioners”, *SIAM J. Sci. Comp. Taylor and Francis,* Vol. 21, pp.1804-1822, 2000.

13. Chow, E, Falgout, R., D., Hu, J, J., Tuminaro, R., S., and Yang,U., M., “A Survey of Parallelization Techniques for Multigrid Solvers”, *Parallel Processing for Scientific Computing, SIAM Series on Software, Environments and Tools,* Chapter 10, 2006.

14. Chu,H., and Hu, B., "A New Comprehensive Model for Nucleate Pool Boiling Heat Transfer of Pure Liquid at Low to High Heat Fluxes including CHF", *Int. J. of Heat and Mass Transfer,* Vol. 52, pp.4203-4210, 2009.

15. Chu, H., and Yu, B., "A New Comprehensive Model for Nucleate Pool Boiling Heat Transfer of Pure Liquid at Low to High Heat Fluxes including CHF", *International Journal of Heat and Mass Transfer*, Vol. 52, pp. 4203-4210, 2009.

16. Cooper, M.G., "Saturation Nucleate Pool Boiling - A Simple Correlation", *Institute of Mechanical Engineers, London, IchemE Symposium Series,* Vol. 86, pp. 786-793, 1984.

17. Cooper, M.G., and Llyod, A.J.P., "The Microlayer in Nucleate Pool Boiling", *Int. J. of Heat and Mass Transfer,* Vol. 12, pp. 895-913, 1969.

18. Costello, C.P. and Frea, W.J., "A salient non-hydrodynamic effect in pool boiling burnout of small semi-cylinder heater.", *Chem. Eng. Prog. Symp.* Series 61, pp. 258-268, 1965.

19. Croce, R., Gribel, M., and Schweitzer, M., A., "A Parallel Level-Set Approach for Two Phase Flow Problems with Surface Tension in Three Space Dimensions" , Preprint 157, Sonderforschungsbereich 611, 2004.

20. Das, A.K., Das, P.K., Saha, P., "Heat Transfer during Pool Boiling based on Evaporation from Micro and Macrolayer", *Int. J. of Heat and Mass Transfer,* Vol. 49, pp.3487-3499, 2006.

21. Darwish, M., Saad, T., and Hamdan, Z., "Parallelization of an Additive Multigrid Solver",*Num. Heat Transfer Part B,* Vol. 54, pp.157-185, 2008.

22. Dhir, V., K., "Burnout in Pool Boiling", Thermopedia.

23. Dhir, V.K., "Boiling Heat Transfer", *Ann. Rev. Fluid Mechanics,* Vol. 30, pp. 365-401. 1998.

24. Dhir, V.K., "Mechanistic Prediction of Nucleate Boiling Heat Transfer - Achievable or a Hopeless Task", *Journal of Heat Transfer,* Vol. 128, pp. 1-12. 2006.

25. Dhir, V.K., and Liaw, S., P., "Framework for a Unified Model for Nucleate and Transition Pool Boiling", *J. Heat Transfer,*Vol. 111, pp.739-745, 1989.

26. Dusan, "On the Spreading of Liquids on Solid Surface: Static and Dynamic Contact Lines", *Annu. Rev. Fluid Mechanics,* Vol. 11, pp. 371-400, 1979.

27. Esmaeeli, A., and Tryggvason, G., "Computationa of Film Boiling, Part I: Numerical Method", *Int. J. of Heat and Mass Transfer,*Vol. 47, pp. 5451-5461, 2004.

28. Fedkiw, R., P., Aslam, T., Merriman, B., Osher S., “A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method)”, *J. Computational Physics*, pp.1-56, 1999.

29. Forster, D.E., and Greif, R., "Heat Transfer to a Boiling Liquid - Mechanism and Correlation", *ASME J. Heat Transfer*, Vol 81, pp. 43-53, 1959.

30. FLUENT tutorial, "Horizontal Film Boiling", May 27, 2005.

31. Gaertner, R.F., "Distribution of Active Sites in the Nucleate Boiling of Liquids", *Chem. Engg. Prog. Symp.,*59 (41), pp. 52-61. 1963.

32. Gaertner, R.F., “Photographic Study of Nucleate Pool Boiling on a Horizontal Surface”, *J. Heat Transfer,* pp. 17-27, 1965.

33. Gaertner, R.F. and Westwater, J.W., “Population of Active Sites in Nucleate Boiling Heat Transfer”, *Chem. Engg. Prog. Symp.,* Vol. 56, pp. 39-61. 1963.

34. Garg, D., and Dhir. V.K.,"Two dimensional numerical simulation of boiling curve". *9th Int. Conf. on Boiling and Cond. Heat Trans.*, Boulder, CO, 2015

35. Gerardi, C., Buongiorno, J., Hu, L., McKrell, T., "Study of Bubble Growth in Water Pool Boiling through Synchronized, Infrared Thermometry and High Speed Video", *Int. J. of Heat and Mass Transfer,*Vol. 53, pp. 4185-4192, 2010.

36. Gibou, F., Chen, L., Nguyen, D., Banerjee, S., “A Level Set based Sharp Interface Method for the Multiphase Incompressible Navier Stokes Equations with Phase Change”,*J. Computational Physics*, 222, pp.536-555, 2006.

37. Golobic, I., Petkovsek, J., Kenning,, D.B.R., "Bubble Growth and Horizontal Coalescence

in Saturated Pool Boiling on a Titanium Foil, Investigated by High Speed IR Thermography", *Int. J. of Heat and Mass Transfer,* Vol. 55, pp. 1385-1402, 2012.

38. Gong, S., Cheng, P., "Lattice Boltmann Simulations for Surface Wettability Effects in Saturated Pool Boiling Heat Transfer", *Int. J. of Heat and Mass Transfer,* Vol. 85, pp. 635-646, 2015.

39. Gramer, L., "Kelvin Helmholtz Instabilites", GFD-II, 2007.

40. Guan, C.K., Klausner, J., F., and Mei, R., "A New Mechanistic Model for Pool Boiling CHF on Horizontal Surfaces", *Int. J. of Heat and Mass Transfer,* Vol. 54, pp.3960-3969, 2011.

41. Ha, S.J., and No, H.C., "A dry spot model for critical heat flux in pool and forced convection boiling", *Int. J. of Heat and Mass Transfer,* Vol. 41, pp.303-311, 1998.

42. Harlow, F., and Welch, J.,E., "Numerical Calculation of Time-Dependent Viscous Incompressible Flows of Fluid with a Free Surface", *Physics of Fluids*, Vol. 8, pp.2182-2189, 1965.

43. Hasegawa, S., Echigo, R., and Takegawa, T., "Maximum heat fluxes for pool boiling on partially ill-wettable heating surfaces", *Bull. JSME 96,* pp. 1076-1084, 1973.

44. He, Y., Shoji, M., Maruyama, S., "Numerical study of high heat flux pool boiling heat transfer", *Int. J. of Heat and Mass Transfer,* Vol. 44, pp.2357-2373, 2001.

45. Hsu, Y.-Y, Graham, R.W., "Transport processes in boiling and two phase systems", American Nuclear Society, Lagrange Park, Illinois, pp. 15-18, 1986.

46. Ishii, M., "Thermo-Fluid Dynamic Theory of Two Phase Flows", Eyrolles Paris, 1975.

47. Jacobsen, D., DeLeon, R., Senocak, I., "Large-Eddy Simulations of Turbulent Incompressible Flow on GPU Cluster", *Comp. Sci Engg.,* Vol. 15, pp. 26-33, 2013.

48. John, V. and Novo, J., "On (essentially) non-oscillatory discretizations of evolutionary convection-diffusion equations", Vol. 231, *J. Comp. Phy,* pp. 1570-1586, 2012.

49. Judd, R.L., and Hwang, K.S., "A Comprehensive Model for Nucleate Boiling Heat Transfer, Including Microlayer Evaporation", *ASME J. Heat Transfer,* Vol. 98, pp. 623-629.

50. Juric, D., and Tryggvason, G., "Computational of Boiling Flows", *Int. J. Multiphase Flow*, vol. 24, pp.387-410,1998.

51. Kalininm E.K., Berlin, I.T., Kostyuk, V.V., and Nosova,E.M., "Heat transfer during transition boiling of cryogenic liquids", *Advances in Cryogenic Engineering*, Vol. 21, pp. 273-277, 1976.

52. Kandlikar, S.G., "A Theroretical Model to Predict Pool Boiling CHF Incorporating Effects of Contact Angle and Orientation",*J. Heat Transfer*, vol. 123, pp.1071-1079,2001.

53. Kocamustafaogullari, G., and Ishii, M., "Interfacial Area and Nucleation Site Density in Boiling Systems", *Int. J. . Heat and Mass Transfer*, vol. 26(9), pp.1377-1387,1983.

54. Klimenko,V.V., "Film Boiling on a Horizontal Plate - New Correlation", *Int. J. of Heat and Mass Transfer,* Vol. 24, pp. 69-79, 1981.

55. Kolev, N.I., "How Accurately Can We Predict Nucleate Boiling", *Exp. Thermal Fluid. Sci.,* Vol. 10, pp. 370-378, 1995.

56. Kunkelmann, C., PhD Thesis, “Numerical Modeling and Investigation of Boiling Phenomena”, 2011.

57. Kutateladze, S.S., “On the Transition to Film Boiling under Natural Convection”, *Kotloturbostroenie*,No. 3, p.10.,1948.

58. Lamb, H., *Hydrodynamics,* Dover Pulications Inc., 6th ed, New York, 1945.

59. Lay, J.H., and Dhir, V.K., "Shape of a Vapor Stem during Nucleate Boiling of Saturated Liquids", *ASME J. Heat Transfer*, Vol. 117, pp. 394-401, 1995.

60. Lee, L.Y.W., Chen. J.C., "Liquid Solid Contact Measurements Using a Surface Thermocouple Temperature Probe in Atmospheric Pool Boiling Water", *Int. J. Heat and Mass Transfer*, 28, pp. 1415-1423, 1985.

61. Lee, R.C, and Nydahl, J.E., "Numerical Calculation of Bubble Growth in Nucleate Boiling from Inception Through Departure", *Transactions of the ASME.,*Vol. 111, pp. 474-479, 1989.

62. Li, D and Dhir, V.K., "Numerical Study of a Single Bubble Sliding on a Downward Facing Heated Surface", *J. Heat Transfer*, Vol. 111, pp. 877-883, 2007.

63. Li, J., Renardy, Y.,Y., and Renardy, M., “Numerical Simulation of Breakup of a Viscous Drop in Simple Shear Flow through a Volume-of-Fluid Method”, *Physics of Fluids*, Vol. 12, pp. 269-282, 2000.

64. Liaw, S.P., Dhir, V.K., "Effect of surface wettability on transition boiling heat transfer from a vertical surface", *Proc. 8th Int. Heat Transfer Conf.* , San Francisco, USA, 1986.

65. Liaw, S.P, and Dhir, V.K., “Void Fraction Measurements during Saturated Pool Boiling of

Water on Partially Wetted Vertical Surfaces", *J. Heat Transfer*, Vol. 111, pp. 731-738, 1989.

66. Lida, Y., and Kobayasi, K., "Distributions of Void Fractions above a Horizontal heating Surface", *Bulletin of JSME*, Vol. 12,No. 50, pp.1076-1084.

67. Lienhard, J.H., and Dhir, V.K., "Extended Hydrodynamic Theory of the Peak and Minimum Pool Boiling Heat Fluxes", NASA Rept., CR-2270, 1973.

68. Luttich, T., Marquardt, W., Buchholz, M., Auracher, H., "Towards a Unifying Heat Transfer Correlation Along the Entire Boiling Curve", *Int. J. Thermal Sciences*, Vol. 43, pp. 1125-1139. 2004.

69. Maracy, M., Winterton, R.H.S., " Hysteresis and Contact Angle Effects in Transition Pool Boiling of Water", *Int. J. Heat Mass Transfer,* 31, pp. 1443-1449,1988.

70. Maruyama, S., Shoji, M., Shimizu, S., "A Numerical Simulation of Transition Boiling Heat Transfer", 1992.

71. Mattor, N., Williams, T.J., Hewett, D., W., "Algorithm for solving tridiagonal matrix problems in parallel", *Parallel Computing* , Vol 21, Issue 11, pp. 1769-1782, 1995.

72. McCaslin, J.O., Courtine, E., Desjardins, O., "A Fast Marching Approach to Multidimensional Extrapolation", *J. Comp. Phy,* Vol. 274, pp. 393-412, 2014.

73. Moin , P., and Mahesh, K., "Direct Numerical Simulation: A Tool in Turbulence Research", *Annual Reviews of Fluid Mechanics*, Vol. 30, pp. 539-578, 1998.

74. Moore, F., D., and Mesler, R., B., "The Measurement of Rapid Surface Temperature Fluctuations during Nucleate Boiling of Water", *AIChE J.,*Vol.7, No.4, pp.620-624, 1966.

75. Mikic, B.B., and Rohsenow, W.M., "A New Correlation of Pool Boiling Data, Including the Effect of Heating Surface Characteristics", *ASME J. Heat Transfer*, Vol. 9, pp. 245-250.

76. Mukherjee, A., and Dhir, V.K., "Study of Lateral Merger of Vapor Bubbles during Nucleate Pool Boiling", *J. Heat Transfer,* Vol. 126, pp. 1023-1039.

77. Nguyen, D., Q., Fedkiw, R., P., and Kang, M., "A Boundary Condition Capturing Method for Incompressible Flame Discontinuities", *J. Computational Physics*, Vol. 172, Issue 1, pp.71-98, 2001.

78. Osher, S., and Fedkiw, R., "Level Set Methods and Dynamic Implicit Surfaces", Springer, 3rd ed., 2002.

79. Ohser, S., and Sethian, J.A., "Front Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations", *J. Computational Physics*, Vol. 79, pp.12-49, 1988.

80. Patanker, S.V, "A Calculation Procedure for Two Dimensional Elliptic Situations", *Numerical Heat Transfer*, Vol. 4, pp. 409-425, 1981.

81. Ragheb, H.S., Cheng. S.C., "Surface wetted area during transition boiling in forced convective flow", J. Heat Transf., 101(2), pp. 381-383, 1979.

82. Ramilison, J.M., Lienhard, J.H., "Transition Boiling Heat Transfer and the Film Transition Regime", *Transactions of the ASME.,* Vol. 109, pp. 746-751, 1987 .

83. Rohsenow, W.M., "A Method of Correlating Heat Transfer Data for Surface Boiling of Liquids", *Trans. ASME,* Vol. 84, p. 969, 1962.

84. Sato, Niceno, "A Sharp-interface phase change model for a Mass conservative interface tracking method", *Journal of Computational Physics,* Vol. 249, pp. 127-161, 2013.

85. Sethian, J.A., “Level Set Methods and Fast Marching Methods”, Cambridge University Press, 1999.

86. Settari, A., and Aziz, K., “A Generalization of the Additive Correction Methods for the Iterative Solution of Matrix Equations”, *SIAM J Numerical Analysis,* Vol. 10, p. 506, 1973.

87. Sielaff, A., Dietl, J., Herbert, S., Stephan, P., "The Influence of System Pressure on Bubble Coalescence in Nucleate Boiling", *Heat Transfer Engineering,*35:5, pp. 420-429, 2014.

88. Shewchuk, J.R., "An Introduction to the Conjugate Gradient Method without the Agonizing Pain", 1994.

89. Shoji,M.,, "Study of Boiling Chaos: A Review", *Int. J. Heat Mass Transfer,* Vol. 47, 1105-1128, 2004.

90. Son, G., and Dhir, V.K., ”Numerical Simulation of Saturated Film Boiling on a Horizontal Surface”, *J. Heat Transfer*, Vol. 119, pp.525-533,1997.

91. Son, G., and Dhir, V.K., ”A Level Set Method for Analysis of Film Boiling on an Immersed Solid Surface, *Numerical Heat Transfer, Part B* ,Vol. 52, pp.153-177, 2007.

92. Son, G., and Dhir, V.K.,”Numerical Simulation of Nucleate Boiling on a Horizontal Surface at High Heat Fluxes”, *Int. J of Heat and Mass Transfer*, Vol. 51, pp.2566-2582, 2008.

93. Son, G., Dhir, V.K., Ramanujapu, N., "Dynamics and Heat Transfer Associated with a Single Bubble during Nucleate Boiling on a Horizontal Surface", *J. Heat Transfer*, Vol.

121, pp. 623-631, 1999.

94. Stephan,K., and Abdelsalam, M., "Heat Transfer Correlations for Natural Convection Boiling", *Int. Journal Heat and Mass Transfer,* Vol. 23, pp. 73-87, 1980.

95. Stephan,K., Fuchs, T., Wagner, E., and Schweizer, N, "Transient Local Heat Fluxes During the Entire Vapor Bubble Lifetime", *Proc. of ECI Int. Conf. on Boiling Heat Transfer,* Florianopolis SC Brazil, May 3-7, 2009.

96. Sultan, M., Judd, R.L, "Spatial Distribution of Active Sites and Bubble Flux Density", *J. Heat Transfer*, 100, pp. 56-62, 1978.

97. Sussman, M., Smereka, P., and Osher, S.J., "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow", *J. Comp. Phy* Vol.114, pp. 146-159, 1994.

98. Sussman, M., Smith, K.M., Hussainin, M.Y., Ohta, M., Wei-Zhi, R., "A Sharp Interface Method for Incompressible Two-Phase Flows", *Journal of Computational Physics,* Vol. 221, pp. 469-505, 2007.

99. Theofanous, T.G., Tu, J.P., Dinh, A.T., and Dinh, T.N., "The Boiling Crisis Phenomenon Part I: Nucleation and Nucleate Boiling Heat Transfer", *Exp. Thermal and Fluid Science,*Vol. 26, pp. 775-792, 2002.

100. Theofanous, T.G., Tu, J.P., Dinh, A.T., and Dinh, T.N., "The Boiling Crisis Phenomenon Part II: Dryout Dynamics and Burnout", *Exp. Thermal and Fluid Science,*Vol. 26, pp. 793-810, 2002.

101. Tomar, G., Biswas G., Sharma A., and Agrawal, A, "Numerical simulation of bubble growth in film boiling using a coupled level-set and volume-of-fluid method", *Physics of Fluids*, Vol 17, 2005.

102. Tong,, L.S., Young, J.D.,"Phenomenological transition and film boiling heat transfer correlation", *Heat Transf*., 4, pp. 120-124, 1974.

103. Wang, C.H., and Dhir, V.K., "Effect of Surface Wettability on Active Nucleation Site Density during Pool Boiling of Water on a Vertical Surface", *J. Heat Transfer,* Vol. 115, pp. 659 - 669, 1993.

104. Winterton, Richard, H.S., "Boiling",Thermopedia.

105. Welch, S.W., "Local Simulation of Two Phase Flows including Interface Tracking with Mass Transfer", *J. Computational Physics*, vol. 121, pp.142-154, 1995.

106. Wu, J., Dhir, V.K., and Qian, J., "Numerical Simulation of Subcooled Nucleate Boiling by Coupling Level-Set Method with Moving-Mesh Method", *Numerical Heat Transfer, Part B,* Vol. 51, pp 535-563, 2007.

107. Wu, J and Dhir, V.K., "Numerical Simulation of Dynamics and Heat Transfer Associated with a Single Bubble in Subcooled Boiling and in the presence of Noncondensables", *J. Heat Transfer,* Vol. 133, p. 041502-1, 2011.

108. Zhao, Y.H., Masuoka, T., and Tsuruta T., "Unified Theoretical Prediction of Fully Developed Nucleate Boiling and Critical Heat Flux Based on a Dynamic Microlayer Model", *Int. J. of Heat and Mass Transfer,*Vol.45, pp.3189-3197, 2002.

109. Zhu, J., Solving Partial Differential Equations on Parallel Computers, *World Scientific Publishing Co. Pvt. Ltd.,*1994.

110. Zuber, N., "Hydrodynamic Aspects of Boiling Heat Transfer", PhD Thesis, University of California, Los Angeles, 1959.

Appendix I - Matrix Solvers

Multigrid Algorithm

Considering a linear PDE - Lu(x)=f(x); where, $L = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

The above equation can be discretized as following:

$$a_{i,j}^{p}\phi_{i,j} = a_{i,j}^{w}\phi_{i-1,j} + a_{i,j}^{e}\phi_{i+1,j} + a_{i,j}^{n}\phi_{i,j+1} + a_{i,j}^{s}\phi_{i,j-1} + b_{i,j}$$

The residual vector is defined as:

$$r_{i,j} = a_{i,j}^{w}\phi_{i-1,j} + a_{i,j}^{e}\phi_{i+1,j} + a_{i,j}^{n}\phi_{i,j+1} + a_{i,j}^{s}\phi_{i,j-1} + b_{i,j} - a_{i,j}^{p}\phi_{i,j}$$

The sum of all the residuals given as below is extended to all the residuals of the block

$$R_{I,J} = \sum r_{i,j} = \sum (b_{i,j} - a_{i,j}^{p}\phi_{i,j}) + a_{i,j}^{w}\phi_{i-1,j} + a_{i,j}^{e}\phi_{i+1,j} + a_{i,j}^{n}\phi_{i,j+1} + a_{i,j}^{s}\phi_{i,j-1} +$$
$$a_{i+1,j}^{w}\phi_{i,j} + a_{i+1,j}^{e}\phi_{i+2,j} + a_{i+1,j}^{n}\phi_{i+1,j+1} + a_{i+1,j}^{s}\phi_{i+1,j-1} +$$
$$a_{i,j+1}^{w}\phi_{i-1,j+1} + a_{i,j+1}^{e}\phi_{i+1,j+1} + a_{i,j+1}^{n}\phi_{i,j+2} + a_{i,j+1}^{s}\phi_{i,j} +$$
$$a_{i+1,j+1}^{w}\phi_{i,j+1} + a_{i+1,j+1}^{e}\phi_{i+2,j+1} + a_{i+1,j+1}^{n}\phi_{i+1,j+2} + a_{i+1,j+1}^{s}\phi_{i+1,j}$$

An additive correction,Φ, restricted to be constant in each block is used to improve the current value of ϕ.

$$\phi_{i,j} \leftarrow \phi_{i,j} + \Phi_{i,j}$$
$$\phi_{i,j} \leftarrow \phi_{i+1,j} + \Phi_{i,j}$$
$$\phi_{i,j} \leftarrow \phi_{i,j+1} + \Phi_{i,j}$$
$$\phi_{i,j} \leftarrow \phi_{i+1,j+1} + \Phi_{i,j}$$

The intention is to impose the block residual after the correction:

$$R_{I,J} = 0$$

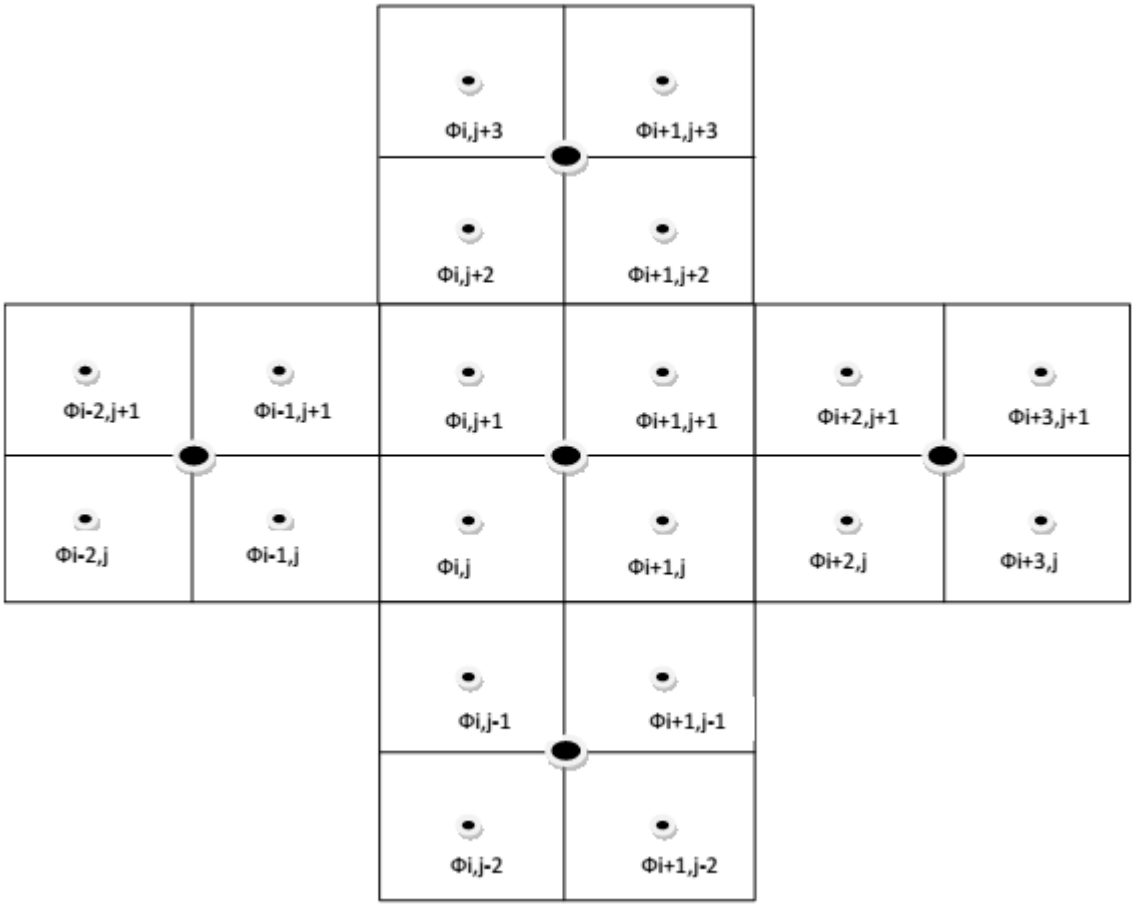


Figure 60 Agglomeration of Cells

$$a_{i,j}^{w}\left(\phi_{i-1,j}+\Phi_{I-1,J}\right)+a_{i,j}^{e}\left(\phi_{i+1,j}+\Phi_{I,J}\right)+a_{i,j}^{n}\left(\phi_{i,j+1}+\Phi_{I,J}\right)+a_{i,j}^{s}\left(\phi_{i,j-1}+\Phi_{I,J-1}\right)+b_{i,j}-a_{i,j}^{p}\left(\phi_{i,j}+\Phi_{I,J}\right)+$$
$$a_{i+1,j}^{w}\left(\phi_{i,j}+\Phi_{I,J}\right)+a_{i+1,j}^{e}\left(\phi_{i+2,j}+\Phi_{I+1,J}\right)+a_{i+1,j}^{n}\left(\phi_{i+1,j+1}+\Phi_{I,J}\right)+a_{i+1,j}^{s}\left(\phi_{i+1,j-1}+\Phi_{I,J-1}\right)+b_{i+1,j}$$
$$-a_{i+1,j}^{p}\left(\phi_{i+1,j}+\Phi_{I,J}\right)+a_{i,j+1}^{w}\left(\phi_{i-1,j+1}+\Phi_{I-1,J}\right)+a_{i,j+1}^{e}\left(\phi_{i+1,j+1}+\Phi_{I,J}\right)+a_{i,j+1}^{n}\left(\phi_{i,j+2}+\Phi_{I,J+1}\right)+a_{i,j+1}^{s}\left(\phi_{i,j}+\Phi_{I,J}\right)$$
$$+b_{i,j+1}-a_{i,j+1}^{p}\left(\phi_{i,j+1}+\Phi_{I,J}\right)+a_{i+1,j+1}^{w}\left(\phi_{i,j+1}+\Phi_{I,J}\right)+a_{i+1,j+1}^{e}\left(\phi_{i+2,j+1}+\Phi_{I+1,J}\right)+a_{i+1,j+1}^{n}\left(\phi_{i+1,j+2}+\Phi_{I,J+1}\right)$$
$$+a_{i+1,j+1}^{s}\left(\phi_{i+1,j}+\Phi_{I,J}\right)+b_{i+1,j+1}-a_{i+1,j+1}^{p}\left(\phi_{i+1,j+1}+\Phi_{I,J}\right)=0$$

The correction equation is thus obtained as follows:

$$A\Phi = B$$

$$A_{I,J}^{p}\Phi_{I,J} = A_{I,J}^{w}\Phi_{I-1,J} + A_{I,J}^{e}\Phi_{I-1,J} + A_{I,J}^{s}\Phi_{I,J-1} + A_{I,J}^{n}\Phi_{I,J+1} + B_{I,J}$$

where

$$A_{I,J}^{p} = \sum a_{i,j}^{p} - \left(a_{i+1,j}^{w} + a_{i+1,j+1}^{w} + a_{i,j}^{e} + a_{i,j+1}^{e} + a_{i,j}^{n} + a_{i+1,j}^{n} + a_{i,j+1}^{s} + a_{i+1,j+1}^{s}\right)$$
$$A_{I,J}^{w} = a_{i,j}^{w} + a_{i,j+1}^{w}$$
$$A_{I,J}^{e} = a_{i+1,j}^{e} + a_{i+1,j+1}^{e}$$
$$A_{I,J}^{s} = a_{i,j}^{s} + a_{i+1,j}^{s}$$
$$A_{I,J}^{n} = a_{i,j+1}^{n} + a_{i+1,j+1}^{n}$$
$$B_{I,J} = \sum r_{i,j}$$

Line by Line Tri Diagonal Matrix Algorithm (LNTDMA)

$$a_{i,j}^{p}\phi_{i,j} = a_{i,j}^{w}\phi_{i-1,j} + a_{i,j}^{e}\phi_{i+1,j} + a_{i,j}^{n}\phi_{i,j+1} + a_{i,j}^{s}\phi_{i,j-1} + b_{i,j}$$

The principle to solve the linear algebraic equation using line-by-line method is to solve for the unknown scalars along one line by the TDMA while using the approximate values of neighboring scalars . The above equation can hence be approximated by:

$$A_i\phi_i = B_i\phi_{i+1} + C_i\phi_{i-1} + D_i$$

In the above formulation, A_i, B_i, C_i and D_i are the known coefficients. Suppose the grid points along the considered line are numbered as 1,2,....,M+1 with 1 and M+1 denoting the boundary points and the range 2,...M constituting the internal points for which the solution has to be found out. With the unknown boundary values getting eliminated:

$$C_2 = 0; \quad B_M = 0$$

TDMA Algorithm is explained as:

$$P_2 = \frac{B_2}{A_2}; \quad Q_2 = \frac{D_2}{A_2}$$

For i = 3,....M, the variables P and Q are calculated as:

$$P_i = \frac{B_i}{A_i - C_i P_{i-1}}; \quad Q_i = \frac{D_i + C_i Q_{i-1}}{A_i - C_i P_{i-1}}$$

With $P_M = 0$ and setting $\phi_M = Q_M$.. The following expression is used to compute $\phi_{M-1, \ldots} \phi_2$ by back substitution.

$$\phi_i = P_i\phi_{i+1} + Q_i$$

Block Correction Procedure

The speed of convergence can further be enhanced by supplementing it with additive-correction method.

$$a_{i,j}^{p}\phi_{i,j} = a_{i,j}^{w}\phi_{i-1,j} + a_{i,j}^{e}\phi_{i+1,j} + a_{i,j}^{n}\phi_{i,j+1} + a_{i,j}^{s}\phi_{i,j-1} + b_{i,j}$$

The above equation is written for all the internal grid points defined by i = 2,...M and j = 2,...N. If the unknown boundary values have been eliminated then:

$$a_{\mathrm{M},j}^{e} = 0,\ a_{2,j}^{w} = 0,\ \ a_{i,\mathrm{N}}^{n} = 0,\ a_{i,2}^{s} = 0$$

The concept behind block correction is that an unconverged field ϕ^* obtained from prior iterations is corrected by adding uniform corrections $\bar{\phi}$ along lines of constant i or j. These corrections satisfy the integral conservation over the control volume blocks defined by lines of constant i or j.

$$A_i\overline{\phi_i} = B_i\overline{\phi_{i+1}} + C_i\overline{\phi_{i-1}} + D_i$$

where,

$$A_i = \sum\left(a_{i,j}^{P} - a_{i,j-1}^{s} - a_{i,j+1}^{n}\right)$$

$$B_i = \sum a_{i,j}^{e}$$

$$C_i = \sum a_{i,j}^{w}$$

$$D_i = \sum\left(a_{i,j}^{e}\phi_{i+1,j}^{*} + a_{i,j}^{w}\phi_{i-1,j}^{*} + a_{i,j+1}^{n}\phi_{i,j+1}^{*} + a_{i,j-1}^{s}\phi_{i,j-1}^{*} + b_{i,j} - a_{i,j}^{P}\phi_{i,j}^{*}\right)$$

The summation of the above equations are taken for J=2,....N if the LNTDMA is being solved for I=2,.....M. Similarly, the procedure can be adopted for block correction on lines of constant J with summation done for I.

Appendix II - Conservation Properties

1. Conservation properties of the numerical model

The present section cross verifies some of the essential conservation properties any numerical technique should possess in order to ensure high fidelity of the numerical results. If there is no vapor in the domain initially and any nucleation occurs then that nucleation event doesn`t influence the flow field at all, as the vapor is placed by removing the liquid artificially and hence the mass and energy content of the computational volume gets altered (Figure 61) as an artifact. During nucleation the liquid along with its superheat is removed from the domain as shown in Figure 62. The energy lost by liquid during a simulation time (t) span depends on initial bubble size taken (Vol_{bub}), average superheat of the thermal layer removed ($\Delta\overline{T}$) and total number of nucleation during runtime (N_t). The vapor gets added artificially without consuming any liquid superheat.

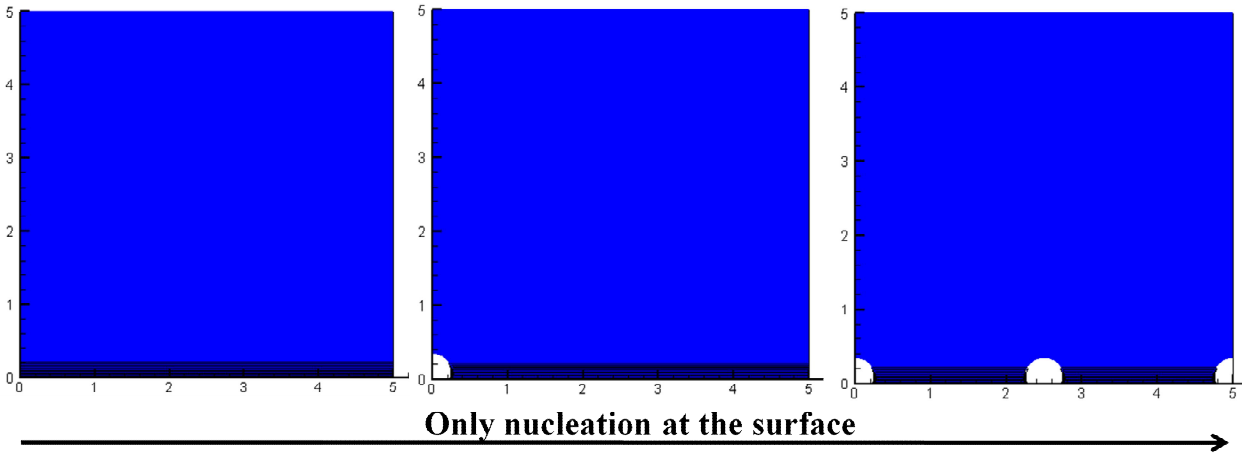


Figure 61 Nucleation event

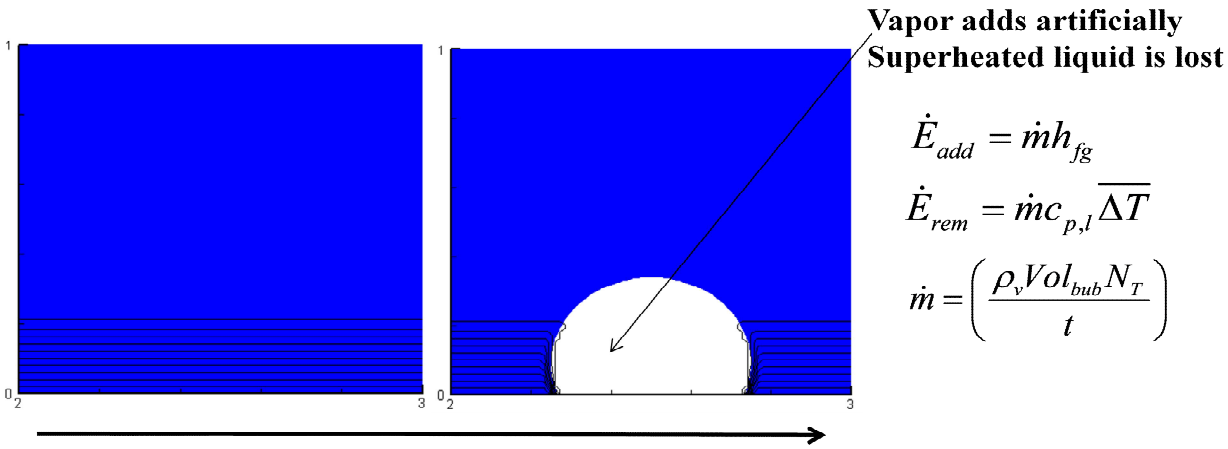


Figure 62 Energy change during nucleation

Under the action of applied wall superheat bubble growth and merger process influences the flow field according to conservation laws. For eg, as the bubble grows, the liquid is pushed out of the domain from the top (Figure 63) in accordance with the principles of mass conservation.

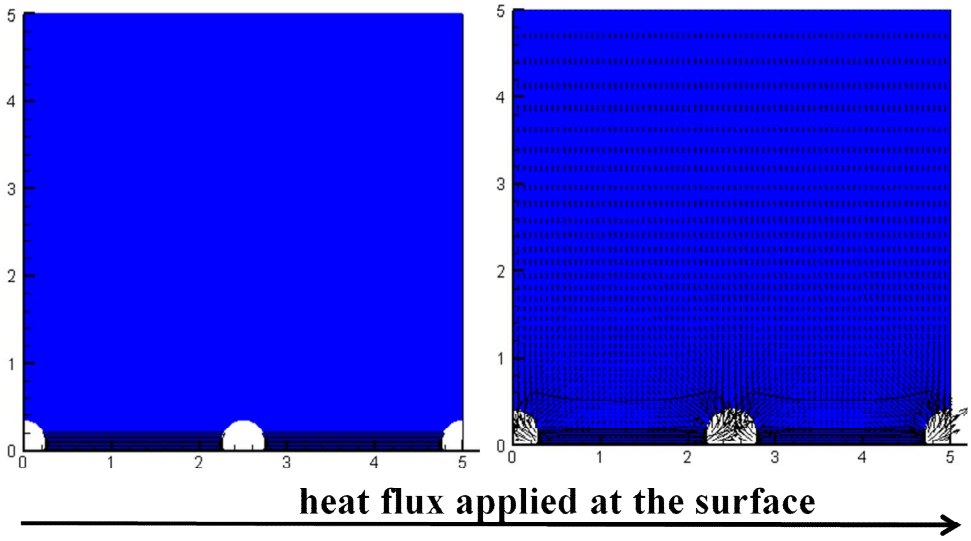


Figure 63 Bubble growth during wall superheated conditions

Estimation related to volume conservation, mass conservation, thermal layer energy conservation and vapor energy conservation are tabulated. For the 2D cases an extruded length of 1 m is assumed for the sake of convenience.

1.1 Mass Conservation

The vapor mass outflow rate from the top of the computational domain gets compensated by the corresponding liquid mass inflow rate to achieve net zero mass flow rate for a constant vapor fraction in the control volume. Net mass imbalance normalized by vapor flow rate out of the control volume during steady state over a computational time step is thereby shown in Table 7. Since the 3D fine grid cases of 38° and 69° contact angle haven`t been run long to observe any vapor escaping the domain, they are not accounted in the present check.

$$\text{Mass flux conservation error} = \left(\frac{\text{Vapor mass flux out - Liquid mass flux in}}{\text{Vapor mass flux out}} \right) \times 100$$

Mass Conservation (ϕ = 38°) 2D Numerical					
ΔT [K]	**Time interval Δt [ms]**	**Time [ms]**	**Vapor Mass Outflow [Kg/s]**	**Liquid Mass Inflow [Kg/s]**	**% Diff Mass Conservation**
20	3.32E-03	222.56	1.29E-03	1.30E-03	-0.3
27.5	7.99E-04	279.2	2.80E-03	2.77E-03	1.32
Mass Conservation (ϕ = 69°) 2D Numerical					
ΔT [K]	**Time interval Δt [ms]**	**Time [ms]**	**Vapor Mass Outflow [Kg/s]**	**Liquid Mass Inflow [Kg/s]**	**% Diff Mass Conservation**
17	3.18E-03	256	5.07E-04	5.00E-04	1.32
22	1.02E-03	267.2	4.86E-04	4.77E-04	1.9
Mass Conservation (ϕ = 38°) 3D Numerical (Coarse)					
ΔT [K]	**Time interval Δt [ms]**	**Time [ms]**	**Vapor Mass Outflow [Kg/s]**	**Liquid Mass Inflow [Kg/s]**	**% Diff Mass Conservation**
20	1.82E-04	46	7.77E-06	7.79E-06	-0.17
27	5.70E-04	62	4.50E-05	4.55E-05	-1.2

Table 7 Mass conservation values

1.2 Volume Conservation

The volume of the computational domain is constant and because of this any vapor bubble that grows during boiling process has to push the liquid out from the top. The difference between the liquid volume flowing out from the domain top, during initial runtime when the vapor hasn`t escaped the domain, and the bubble growth volume normalized by the bubble growth volume is shown in Table 8 and is expressed as:

$$\text{Volume conservation error} = \left(1 - \frac{\text{Volume of liquid pushed out}}{\text{Increase in vapor volume}}\right) \times 100$$

Volume conservation (ϕ = 38°) 2D Numerical					
ΔT [K]	**Time int. Δt [ms]**	**Time [ms]**	**Vap. volume increase [m^3]**	**Liq. volume outflow [m^3]**	**% Diff. volume conservation**
20	3.81E-03	19.20	7.54E-09	7.22E-09	4.3
27.5	5.40E-04	68.80	3.49E-09	3.38E-09	3.2
Volume conservation (ϕ = 69°) 2D Numerical					
ΔT [K]	**Time int. Δt [ms]**	**Time [ms]**	**Vap. volume increase [m^3]**	**Liq. volume outflow [m^3]**	**% Diff. volume conservation**
17	1.72E-03	21.3	1.96E-09	2.02E-09	-3.1
22	2.48E-03	45.1	4.03E-09	3.77E-09	6.42
Volume conservation (ϕ = 38°) 3D Numerical (Coarse)					
ΔT [K]	**Time int. Δt [ms]**	**Time [ms]**	**Vap. volume increase [m^3]**	**Liq. volume outflow [m^3]**	**% Diff. volume conservation**
20	4.51E-04	1.6	8.78E-13	9.43E-13	-7.42
27	2.31E-05	2.4	6.72E-13	6.20E-13	7.65
Volume conservation (ϕ = 38°) 3D Numerical (Fine)					
ΔT [K]	**Time int. Δt [ms]**	**Time [ms]**	**Vap. volume increase [m^3]**	**Liq. volume outflow [m^3]**	**% Diff. volume conservation**
15	2.23E-04	4.8	3.94E-13	3.58E-13	9.1
20	2.44E-04	9.5	8.78E-13	9.34E-13	-6.4
Volume conservation (ϕ = 69°) 3D Numerical (Fine)					
ΔT [K]	**Time int. Δt [ms]**	**Time [ms]**	**Vap. volume increase [m^3]**	**Liq. volume outflow [m^3]**	**% Diff. volume conservation**
15	9.38E-05	3.2	5.06E-13	4.97E-13	1.75
20	3.49E-04	15.8	1.00E-12	1.03E-12	-3.72

Table 8 Volume conservation calculation

1.3 Energy balance at the wall

The energy being provided from the wall goes into dry area (vapor), microlayer and wet area (liquid) as shown in

Figure **64** (sample calculations shown in Table 9). The energy transferred to dry area has been estimated to be less than 1 [W/cm^2]

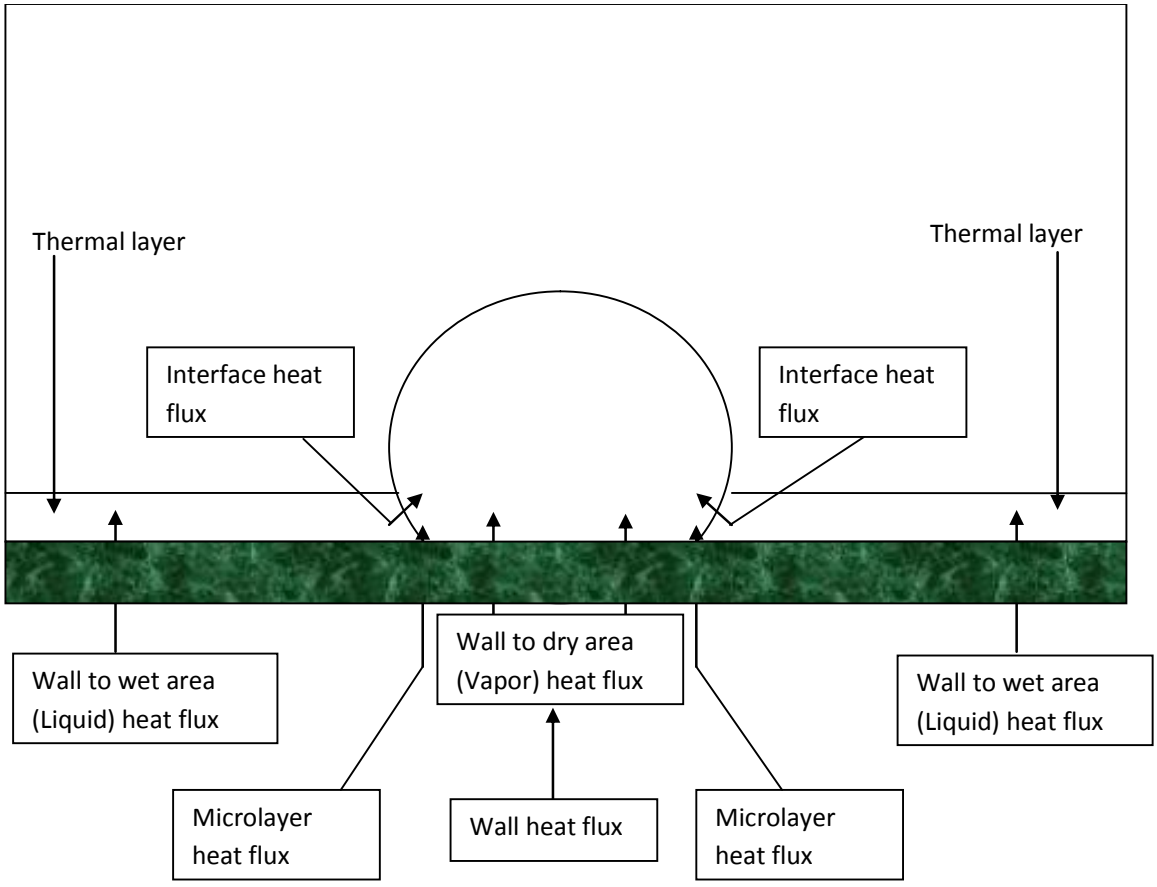


Figure 64 Energy transfer from wall into liquid and vapor

Energy utilized for vapor production = Microlayer heat transfer + Interface heat transfer + Energy utilized for superheating the vapor

Energy utilized for superheating the liquid = Wall heat flux - Energy utilized for vapor production

Wall heat flux partitioning (ϕ = 38°) 2D Numerical							
ΔT [K]	**Time [ms]**	**Time interval [ms]**	**Wall heat flux [W/cm^2]**	**Microlayer heat flux [W/cm^2]**	**Wall to wet area [W/cm2]**	**Wall to dry area [W/cm2]**	**Interface heat flux [W/cm^2]**
20	90	3.82E-04	41.41	8.65	32.76	0	13.96
27.5	120	5.00E-05	78.92	39.37	39.55	0	9.38
Wall heat flux partitioning (ϕ = 69°) 2D Numerical							
ΔT [K]	**Time [ms]**	**Time interval [ms]**	**Wall heat flux [W/cm^2]**	**Microlayer heat flux [W/cm^2]**	**Wall to wet area [W/cm2]**	**Wall to dry area [W/cm2]**	**Interface heat flux [W/cm^2]**
17	192	3.76E-03	19.49	3.8	15.69	0	8.23
22	160	3.58E-03	22.83	4.78	18.05	0	10.01
Wall heat flux partitioning (ϕ = 38°) 3D Numerical (Coarse)							
ΔT [K]	**Time [ms]**	**Time interval [ms]**	**Wall heat flux [W/cm^2]**	**Microlayer heat flux [W/cm^2]**	**Wall to wet area [W/cm2]**	**Wall to dry area [W/cm2]**	**Interface heat flux [W/cm^2]**
20	50	6.64E-04	21.54	1.74	18.8	< 1	1.36
27	60	1.47E-03	60.34	20.57	38.77	< 1	13.99
Wall heat flux partitioning (ϕ = 38°) 3D Numerical (Fine)							
ΔT [K]	**Time [ms]**	**Time interval [ms]**	**Wall heat flux [W/cm^2]**	**Microlayer heat flux [W/cm^2]**	**Wall to wet area [W/cm2]**	**Wall to dry area [W/cm2]**	**Interface heat flux [W/cm^2]**
15	15	1.39E-03	12.71	1.17	10.54	< 1	5.5
20	5	4.70E-03	24.31	5.81	17.5	< 1	10.53
Wall heat flux partitioning (ϕ = 69°) 3D Numerical (Fine)							
ΔT [K]	**Time [ms]**	**Time interval [ms]**	**Wall heat flux [W/cm^2]**	**Microlayer heat flux [W/cm^2]**	**Wall to wet area [W/cm2]**	**Wall to dry area [W/cm2]**	**Interface heat flux [W/cm^2]**
15	10	8.22E-04	16.37	1.87	13.5	< 1	6.7
20	13	7.72E-04	30.59	4.85	24.74	< 1	13.8

Table 9 Wall heat flux partitioning

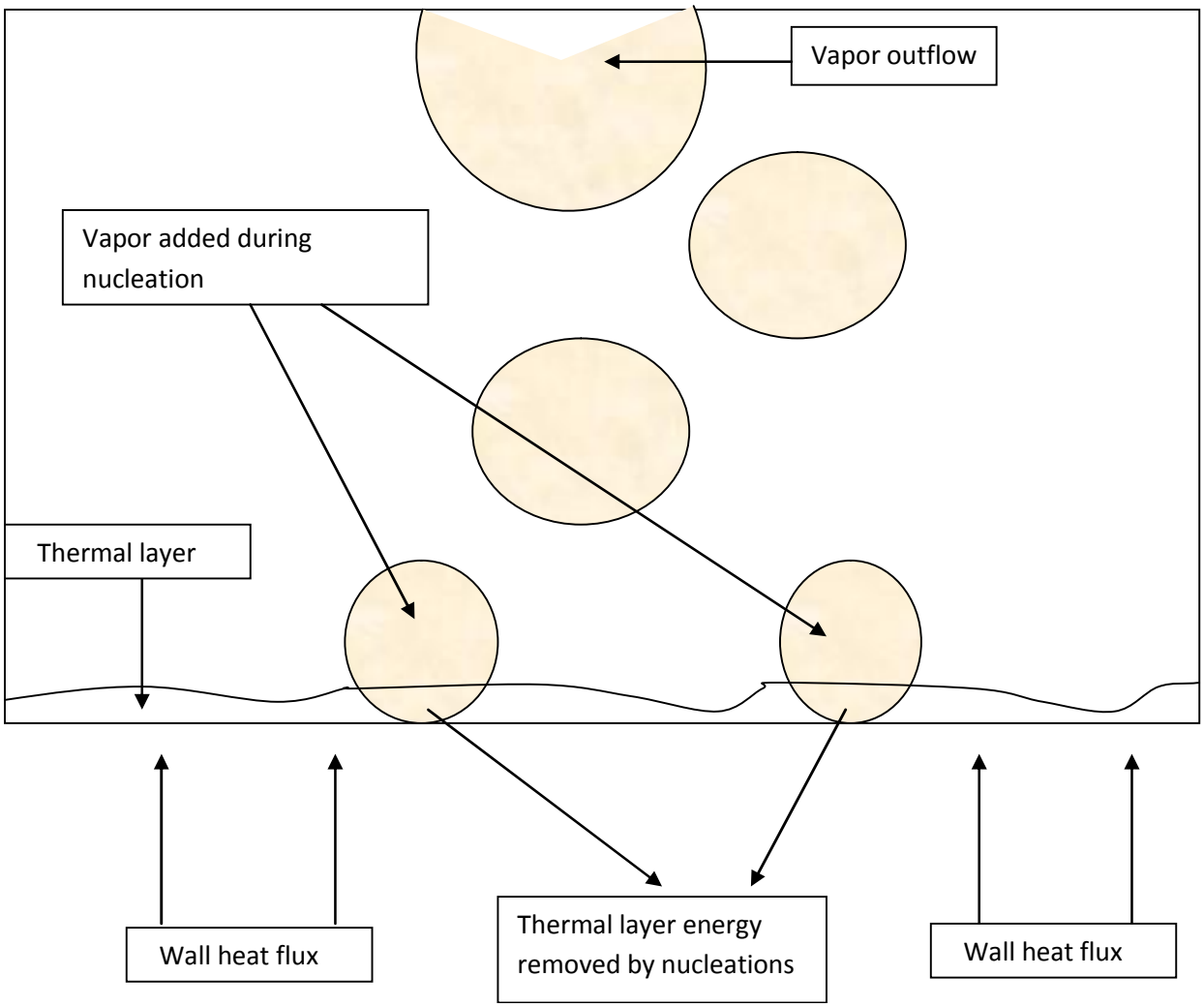


Figure 65 Energy transfer for the present numerical model

1.4 Thermal layer energy conservation

The thermal layer gets energy from the wall and provides energy to the bubble through the interface. As per the current numerical model during nucleation vapor gets added artificially removing liquid along with thermal layer from its original place (shown in Figure **65**). The energy balance of the thermal layer normalized by time step and heater surface area is documented in Table **10 (**time interval is shown in Table **9)** and expressed as:

$$\%\ \text{Difference in thermal layer energy conservation}$$

$$= \left(1 - \frac{\text{Thermal layer energy rate lost during nucleation per unit wall area}}{\text{Increase in thermal layer energy rate per unit wall area}}\right) \times 100$$

where,

Increase in thermal energy rate per unit wall area
= Wall to wet area energy transfer rate per unit wall area
- Interface energy transfer rate per unit wall area

Thermal layer energy conservation (ϕ = 38°) 2D Numerical						
ΔT [K]	**Time [ms]**	**Wall to wet area [W/cm2]**	**Liquid to vapor through interface [W/cm2]**	**Thermal layer energy lost during nucleation [W/cm2]**	**Increase in thermal layer energy [W/cm2]**	**% Diff. in thermal layer energy conservation**
20	90	32.76	13.96	17.89	18.8	4.84
27.5	120	39.55	9.38	26	30.17	13.82
Thermal layer energy conservation (ϕ = 69°) 2D Numerical						
ΔT [K]	**Time [ms]**	**Wall to wet area [W/cm2]**	**Liquid to vapor through interface [W/cm2]**	**Thermal layer energy lost during nucleation [W/cm2]**	**Increase in thermal layer energy [W/cm2]**	**% Diff. in thermal layer energy conservation**
17	192	15.69	8.23	6.2	7.46	16.89
22	160	18.05	10.01	9.39	8.04	-16.79
Thermal layer energy conservation (ϕ = 38°) 3D Numerical (Coarse)						
ΔT [K]	**Time [ms]**	**Wall to wet area [W/cm2]**	**Liquid to vapor through interface [W/cm2]**	**Thermal layer energy lost during nucleation [W/cm2]**	**Increase in thermal layer energy [W/cm2]**	**% Diff. in thermal layer energy conservation**
20	50	18.8	1.36	15.95	17.44	8.54
27	60	38.77	13.99	27.58	24.78	-11.30
Thermal layer energy conservation (ϕ = 38°) 3D Numerical (Fine)						
ΔT [K]	**Time [ms]**	**Wall to wet area [W/cm2]**	**Liquid to vapor through interface [W/cm2]**	**Thermal layer energy lost during nucleation [W/cm2]**	**Increase in thermal layer energy [W/cm2]**	**% Diff. in thermal layer energy conservation**
15	15	10.54	5.5	5.58	5.04	-10.71
20	5	17.5	10.53	7.65	6.97	-9.76
Thermal layer energy conservation (ϕ = 69°) 3D Numerical (Fine)						
ΔT [K]	**Time [ms]**	**Wall to wet area [W/cm2]**	**Liquid to vapor through interface [W/cm2]**	**Thermal layer energy lost during nucleation [W/cm2]**	**Increase in thermal layer energy [W/cm2]**	**% Diff. in thermal layer energy conservation**
15	10	13.5	6.7	8	6.8	-17.65
20	13	24.74	13.8	13.1	10.94	-19.74

Table 10 Thermal layer energy (normalized by time and area) conservation

1.5 Vapor energy conservation

The vapor is generated at the liquid vapor interface, microlayer and every time a nucleation site gets active a small bubble embryo gets added to the computational domain. Vapor production due to latter becomes significant if the initial bubble embryo size is large and the error gets exacerbated if there are multiple cavities nucleating with high frequency. The vapor energy rate conservation error calculated over a computational time step and normalized by heater surface area is shown in Table **11 (**time interval is shown in Table **9)**. The vapor volume change in the computational domain is non-zero only for the 3D fine grid cases as vapor didn`t escape from the top while for the other case vapor configuration is approximately steady state (constant vapor fraction in the computational domain) for the present calculation.

$$\text{Vapor energy rate conservation error} = \left(\frac{\left(\dot{E}_{vap,out} + \dot{E}_{vap,ch}\right) - \left(\dot{E}_{vap,in} + \dot{E}_{vap,add}\right)}{\dot{E}_{vap,out} + \dot{E}_{vap,ch}} \right) \times 100$$

where,

Energy rate going into vapor per unit wall area
(neglecting wall to dry area)

= Microlayer energy rate per unit wall area + Interface energy rate per unit wall area

$\dot{E}_{vap,ch}$ = Vapor energy rate change due to change in vapor volume per unit wall area

$\dot{E}_{vap,in}$ = Energy rate going into vapor per unit wall area

$\dot{E}_{vap,out}$ = Vapor outflow energy rate per unit area

$\dot{E}_{vap,add}$ = Vapor energy rate due to nucleation per unit area

Vapor energy conservation (ϕ = 38°) 2D Numerical						
ΔT [K]	**Time [ms]**	**$\dot{E}_{vap,in}$ [W/cm^2]**	**$\dot{E}_{vap,add}$ [W/cm2]**	**$\dot{E}_{vap,ch}$ [W/cm2]**	**$\dot{E}_{vap,out}$ [W/cm2]**	**% Diff. in vapor energy conservation**
20	90	22.61	≈ 0	≈ 0	23	1.70
27.5	120	48.75	≈ 0	≈ 0	49.91	2.30
Vapor energy conservation (ϕ = 69°) 2D Numerical						
ΔT [K]	**Time [ms]**	**$\dot{E}_{vap,in}$ [W/cm^2]**	**$\dot{E}_{vap,add}$ [W/cm2]**	**$\dot{E}_{vap,ch}$ [W/cm2]**	**$\dot{E}_{vap,out}$ [W/cm2]**	**% Diff. in vapor energy conservation**
17	192	12.03	≈ 0	≈ 0	12.45	3.37
22	160	14.79	≈ 0	≈ 0	13.44	-10.04
Vapor energy conservation (ϕ = 38°) 3D Numerical (Coarse)						
ΔT [K]	**Time [ms]**	**$\dot{E}_{vap,in}$ [W/cm^2]**	**$\dot{E}_{vap,add}$ [W/cm2]**	**$\dot{E}_{vap,ch}$ [W/cm2]**	**$\dot{E}_{vap,out}$ [W/cm2]**	**% Diff. in vapor energy conservation**
20	50	3.1	5.05	≈ 0	10.89	25.16
27	60	34.56	18	≈ 0	63.04	16.62
Vapor energy conservation (ϕ = 38°) 3D Numerical (Fine)						
ΔT [K]	**Time [ms]**	**$\dot{E}_{vap,in}$ [W/cm^2]**	**$\dot{E}_{vap,add}$ [W/cm2]**	**$\dot{E}_{vap,ch}$ [W/cm2]**	**$\dot{E}_{vap,out}$ [W/cm2]**	**% Diff. in vapor energy conservation**
15	15	6.67	≈ 0	7.31	0	8.76
20	5	16.34	≈ 0	15.87	0	-2.96
Vapor energy conservation (ϕ = 69°) 3D Numerical (Fine)						
ΔT [K]	**Time [ms]**	**$\dot{E}_{vap,in}$ [W/cm^2]**	**$\dot{E}_{vap,add}$ [W/cm2]**	**$\dot{E}_{vap,ch}$ [W/cm2]**	**$\dot{E}_{vap,out}$ [W/cm2]**	**% Diff. in vapor energy conservation**
15	10	8.57	≈ 0	8.74	0	1.95
20	13	18.65	≈ 0	18.16	0	-2.70

Table 11 Vapor energy (normalized by time and area) conservation

Appendix III - 3D Parallel Code

```
!**********************************************************************
PROGRAM MAIN
!**********************************************************************
  USE PARAMETERS
  USE ICHOS
  USE PROPS
  USE INDMO
  USE INDDT
  USE IGRID
  USE MUGRD
  USE IBNDS
  USE GRID1
  USE GRIDB
  USE GRIDL
  USE CHCFL
  USE IDREF
  USE UVTBC
  USE IBUBS
  USE LVFBC
  USE QINTB
  USE SPERR
  USE LEVFN
  USE ULVLL
  USE FFSOR
  USE UVTPB
  USE RESLT
  USE UVTPP
  USE CALC1
  USE DIMEN
  USE MPIVAR
  IMPLICIT NONE
  REAL*8 :: AL0,ALPL,BPMAX,BUMAX,BVMAX,BWMAX,RGRAV,RH0,RHOL,RHOV
  REAL*8 ::
CAPL,CAPV,CONDL,CONDV,F0,F1,GRAV,H0,HBOT,HBOT1,HFG,HTOP1,PEL,PMAX,PRL,RAL0,RCV,
RCV0,REL,REV
  REAL*8 ::
RMU,RRE,RRHO,RRR,RU0,SURF,TWAIT,U0,UMAX,VMAX,WMAX,VBOT,VTOP,VISL,VISV,X1,XL,Y0,Y
1,YL,ZL
  INTEGER ::
I,J,K,IACT,IBUB0,IFB,IND,IORDER,IREIN,JI3OUT,JINOUT,JREOUT,JPLOUT,JFLOUT,LTIME,N
!**********************************************************************************************************
CALL MULTICORE
CALL ALLOT
CALL FRAGMENT    ! Parallel decomposition
!**********************************************************************
PI=ACOS(-1.)
CFO=0.5
  IINF=1
  IOUT=2
```

```
  ISYM=3
  IWAL=4
  IWQQ=5                ! Constants
!---------------------------------------------------------------------
  ICONT=1
  ICONB=0
!---------------------------------------------------------------------
  IENO=0
  IENOT=1
  IORDER=1
!---------------------------------------------------------------------
  ITIME=0
  TIME=0.
  ICF=1
JCF=1
  KCF=1
  I2B=1
  N2B=2
  NBUB(1)=0
  NBUB(2)=0
  V0H(1)=1.
  V1H(1)=1.
  V0H(2)=0.
  V1H(2)=-1.
  IREIN=0
  ITSOLV=1
  JMOVE=0
!---------------------------------------------------------------------
  QMIC0=0.
  TINT=0.
  TWAL=1.
  DTWS=0.
  HFGI=0.
  VBUB=0.
  TBCW=1.
  TBCE=1.
  TBCS=1.
  TBCN=1.
  TBCB=1.
  TBCT=1.
!---------------------------------------------------------------------- ! Read the input file
OPEN( 1,FILE='tpi')
READ(1,*)
READ(1,*) IND,LTIME,NI,NJ,NK,IR,MULT
READ(1,*)
READ(1,*) DT0,IUGRD,XL,DL,JMOVE,IENO,IENOT
READ(1,*)
READ(1,*) IGRAV,IBOIL,IQ_LIQ,IQ_VAP,IBODY,ICONT,ICONB
READ(1,*)
READ(1,*) RCV0,YBUB0,XBND,YBND,ZBND,RBND,ESMAX,EFMAX
READ(1,*)
READ(1,*) H0,UIN0,QMIC0
READ(1,*)
```

```
READ(1,*) I69,JMONT,JMONIT,JINOUT,JI3OUT,JPLOUT,JREOUT,JFLOUT
READ(1,*)
READ(1,*) IREF,JREF,KREF
READ(1,*)
READ(1,*)
READ(1,*) IBCW,IBCE,IBCS,IBCN,IBCB,IBCT
READ(1,*)
READ(1,*) CANGMAX,CANGMIN,CANBMAX,CANBMIN
IF(IBOIL.EQ.1) THEN
READ(1,*)
READ(1,*) RHOL,RHOV,VISL,VISV,CONDL,CONDV
READ(1,*)
READ(1,*) CAPL,CAPV,SURF,RGRAV,DTWS,BT0,HFG
ELSE
READ(1,*)
READ(1,*) RO1,RO2,VS1,VS2,SFT0,TK1,TK2
ENDIF
READ(1,*)
READ(1,*) NCAV,TWAIT
READ(1,*)
READ(1,*) ( XCAV(I),I=1,NCAV)
READ(1,*) ( ZCAV(I),I=1,NCAV)
READ(1,*) (TIMEW(I),I=1,NCAV)
CLOSE(1)
DT=DT0
I23=3
IF(NK.LE.3) I23=2
K11=1
KNK=NK
IF(I23.EQ.2) THEN
NK=3
K11=2
KNK=2
KREF=2
IBCB=ISYM
IBCT=ISYM
FZ1=0.
ENDIF

IF(IUGRD.EQ.1) THEN
XU(2)=0.
YV(2)=0.
ZW(2)=0.
DL=XL/(NI-2)
DO 12 I=3,NI
12      XU(I)=XU(I-1)+DL
DO 14 J=3,NJ
14      YV(J)=YV(J-1)+DL
DO 16 K=3,NK
16      ZW(K)=ZW(K-1)+DL
ELSE
OPEN(1,FILE='grd')
READ(1,*)
```

```
DO 22 I=2,NI
22      READ(1,*) XU(I)
READ(1,*)
DO 24 J=2,NJ
24      READ(1,*) YV(J)
IF(I23.EQ.3) THEN
READ(1,*)
DO 26 K=2,NK
26      READ(1,*) ZW(K)
ENDIF
CLOSE(1)
ENDIF
IF(I23.EQ.2) ZW(3)=ZW(2)+1.

XL=XU(NI)
YL=YV(NJ)
ZL=ZW(NK)

IF (TASKID.EQ.0) THEN            !ONLY MASTER PROCESSOR DISPLAYS THIS.
WRITE(6,'(5X,A,6(1PE12.4))') ' X0, Y0, Z0 =',XU( 2),YV( 2),ZW( 2)
WRITE(6,'(5X,A,6(1PE12.4))') ' XL, YL, ZL =',XU(NI),YV(NJ),ZW(NK)
ENDIF

IF(I69 .EQ. 9) OPEN( 9,FILE='mon')
IF(IBOIL.EQ.0) GOTO 50
H0=H0*DTWS**(-1./3.)
GRAV=9.807
AL0=SQRT(SURF/GRAV/(RHOL-RHOV))
U0=SQRT(GRAV*AL0)
REL=U0*AL0/VISL
ALPL=CONDL/RHOL/CAPL
H0=H0*7.14*(VISL*ALPL/GRAV/BT0)**(1./3.)/AL0

IF (TASKID.EQ.0) THEN
WRITE(6,'(A,4(1PE12.4))') '    AL0, U0, H0, RE=',AL0,U0,H0,REL
ENDIF

RAL0=RGRAV**(-1./2.)
RU0=SQRT(RGRAV*RAL0)
RH0=RGRAV**(-1./3.)/RAL0
RRE=RAL0*RU0
H0=H0*RH0
REL=REL*RRE

IF (TASKID.EQ.0) THEN
WRITE(6,'(A,4(1PE12.4))') ' RAL0,RU0,RH0,RRE=',RAL0,RU0,RH0,RRE
ENDIF

RRHO=RHOV/RHOL
RMU=RRHO*VISV/VISL
REV=REL/RMU
SFT0=(RHOL-RHOV)/RHOL
PRL=CAPL*VISL*RHOL/CONDL
```

```
PEL=REL*PRL
HFGI=CAPL*DTWS/HFG

RO1=RRHO
RO2=1.
VS1=REL/RMU
VS2=REL
TK1=PEL/(CONDV/CONDL)
TK2=PEL
RC1=RO1*CAPV/CAPL
RC2=1.
50      CONTINUE

EPS=1.E-5
HEPS=1.E-2
VFG=1./RO1-1./RO2

!--------------                  !MPI Functions
CALL FRAGMENT
CALL GHBUFFER
CALL BUFFER
ALLOCATE(GHVAR(IIST-2:IIEND+2,JJST-2:JJEND+2,KKST-2:KKEND+2,1:2),STAT=IERR)
IF (IERR /=0) PRINT*,'ERROR IN GHVAR BY',TASKID
!--------------
CALL INIT
!--------------------------------
ARO=0.5*(RU(NI)+RU(2))*XL
IF(I23.EQ.3) ARO=XL*ZL

!---------------------------------- Contact Angle
CANGMAX=CANGMAX*PI/180.
CANGMIN=CANGMIN*PI/180.
CANBMAX=CANBMAX*PI/180.
CANBMIN=CANBMIN*PI/180.
COSMIN=COS(CANGMAX)
COSMAX=COS(CANGMIN)
COBMIN=COS(CANBMAX)
COBMAX=COS(CANBMIN)
!----------------------------------------
Y0=YBUB0
IF(YBUB0.LT.-10.) Y0=RCV0*COSMIN

IF (TASKID.EQ.0) THEN
IF(IBOIL.EQ.1) THEN
WRITE(6,'(A,4(1PE12.4))') ' DL,YL,RCV0,QMIC0=',DL,YL,RCV0,QMIC0
ENDIF
ENDIF

IF(IBODY.EQ.1) THEN
DO 70 K=K11,KNK
DO 70 J=1,NJ
DO 70 I=1,NI
!IF(I23.EQ.3) FZ1=(ZP(K)-ZBND)**2
```

```
IF(I23.EQ.3) FZ1=0.
70      S(I,J,K)=SQRT((XP(I)-XBND)**2+(YP(J)-YBND)**2+FZ1)-RBND
CALL SUVW_GET

IF (TASKID.EQ.0) THEN
OPEN(90,FILE='bnd0')
DO 72 IFB=-9,2
F0=IFB*XD(2)
72      CALL BNDXY_PLOT(2,F0)
CLOSE(90)

OPEN(90,FILE='b2xy')
CALL BNDXY_PLOT(2,0.)
CLOSE(90)

IF(I23.EQ.3) THEN
OPEN(90,FILE='b2zy')
CALL BNDZY_PLOT(2,0.)
CLOSE(90)
ENDIF
ENDIF
ENDIF

DO 100 N=1,NCAV
FMIN(N)=1.E6
DO 110 K=2,NKM
DO 110 I=2,NIM
X1=XP(I)-XCAV(N)
Y1=YP(2)-Y0
IF(I23.EQ.3) FZ1=(ZP(K)-ZCAV(N))**2
F1=SQRT(X1**2+Y1**2+FZ1)
IF(F1 .GT. FMIN(N)) GOTO 110
FMIN(N)=F1
ICAV(N)=I
JCAV(N)=2
KCAV(N)=K
110     CONTINUE
IF (TASKID.EQ.0) THEN
WRITE(6,'(A,I3,3(I4),1PE12.4)') ' Active Cavity: N,I,J,K,DF=',N,ICAV(N),JCAV(N),KCAV(N),FMIN(N)
ENDIF
100     CONTINUE

DO 115 N=1,NCAV
IBCAV(N)=0
IF(TIMEW(N).LE.TWAIT) GOTO 115
NBUB(1)=NBUB(1)+1
IBCAV(N)=NBUB(1)
TIMEW(N)=0.
115     CONTINUE

IF(IND .EQ. 0) THEN
!----------------------------------------
IACT=0
```

```
DO 120 N=1,NCAV
IF(IBCAV(N).EQ.0) GOTO 120
IACT=IACT+1

IF(IACT.EQ.1) THEN
DO 122 K=K11,KNK
DO 122 J=1,NJ
DO 122 I=1,NI
IBUB(I,J,K,1)=0
F(I,J,K)=1.E6
122     DDF(I,J,K)=DL
ENDIF

DO 124 K=K11,KNK
DO 124 J=1,NJ
DO 124 I=1,NI
CC(I,J,K)=    F(I,J,K)
SS(I,J,K)=DDF(I,J,K)
IF(I23.EQ.3) FZ1=(ZP(K)-ZCAV(N))**2
IF(I23.EQ.3 .AND. IBODY.EQ.1) FZ1=0.
RCV=RCV0
IF(IBODY.EQ.1 .AND. I23.EQ.3) THEN
RCV=RCV0+(RCV0-RBND)*0.1*COS(PI*ZP(K)/ZL)
ENDIF
124     F(I,J,K)=SQRT((XP(I)-XCAV(N))**2+(YP(J)-Y0)**2+FZ1)-RCV
CALL DDF_GET

DO 126 K=K11,KNK
DO 126 J=1,NJ
DO 126 I=1,NI
IF(CC(I,J,K).LE.SS(I,J,K) .AND. F(I,J,K).LE.DDF(I,J,K)) THEN
IF (TASKID.EQ.0) THEN
WRITE(I69,*) ' Bubble Merged initially at N,I,K=',N,I,K
ENDIF
STOP
ENDIF
IF(F(I,J,K).LT.DDF(I,J,K)) IBUB(I,J,K,1)=IBCAV(N)
F(I,J,K)=AMIN1(CC(I,J,K),F(I,J,K))
126     CONTINUE
CALL DDF_GET
120     CONTINUE
DO 150 K=K11,KNK
DO 150 J=1,NJ
DO 150 I=1,NI
T(I,J,K)=AMAX1(0.,1.-YP(J)/H0)
IF(IGRAV.EQ.0) T(I,J,K)=1.
F1=F(I,J,K)/(2.*DDF(I,J,K))
F1=AMAX1(0.,AMIN1(1.,F1))
IF(F(I,J,K).GE.0.) T(I,J,K)=F1*(2.-F1)*T(I,J,K)

IF(IBODY.EQ.1) THEN
RRR=SQRT(XP(I)**2+(YP(J)-Y0)**2)-RBND
RCV=RCV0
```

```
IF(IBODY.EQ.1 .AND. I23.EQ.3) THEN
RCV=RCV0+(RCV0-RBND)*0.1*COS(PI*ZP(K)/ZL)
ENDIF
T(I,J,K)=AMAX1(0.,1.-RRR/(RCV-RBND))
ENDIF
150     CONTINUE
ENDIF

IF(IND .NE. 0) CALL RESTF_IN
DO 160 K=K11,KNK
DO 160 J=1,NJ
DO 160 I=1,NI
160     BF(I,J,K)=F(I,J,K)

CALL F_BC
CALL H_GET
BDEFF=DEFF
CALL T_BC
CALL U_BC
CALL V_BC
CALL W_BC

I2B=2
IF(IND.EQ.0) CALL IBUB_INIT
CALL IPBUB_GET
CALL BVOLB_GET
BUMAX=1.E-10
BVMAX=1.E-10
BWMAX=1.E-10
BPMAX=1.E-10
DO 310 K=K11,KNK
DO 310 J=1,NJ
DO 310 I=1,NI
BUMAX=AMAX1(BUMAX,ABS(U(I,J,K)))
BVMAX=AMAX1(BVMAX,ABS(V(I,J,K)))
BWMAX=AMAX1(BWMAX,ABS(W(I,J,K)))
310     BPMAX=AMAX1(BPMAX,ABS(P(I,J,K)))

CALL INTP_GET(HTOP, 2,2,0,2)
CALL INTP_GET(HBOT,-2,2,0,2)
CALL N_GET
CALL RK_GET
CALL AM_GET
CALL FO_BC
CALL INT2D_OUT
CALL INT3D_OUT
DO 320 K=2,NKM
DO 320 J=2,NJM
DO 320 I=2,NIM
320     BP(I,J,K)=P(I,J,K)
WRITE(I69,4021) 1+ITIME, TIME,DT
IF(I69 .EQ. 9) CLOSE(9)
!---------------------
```

```
!CALL DEBUFFER
!CALL DEGHBUFFER
!DEALLOCATE(GHVAR,STAT=IERR)
!IF (IERR /=0) PRINT*,'ERROR IN DEALLOCATE GHVAR BY',TASKID
!-------------------------

!CALL PLOTF_OUT(JPLOUT)

!IF (TASKID.EQ.0) THEN
!CALL OFILE_OUT(JFLOUT)
!ENDIF

!************************************************************************
DO 500 ITIME=1+ITIME,LTIME
!************************************************************************
IF (TASKID.EQ.0) THEN
print*,'TASKID=',TASKID,'ITIME',ITIME
if (ITIME.eq.ITIME/10*10) write(*,*) 'TASKID=',TASKID,'ITIME=',ITIME,
F(25,25,25),P(10,50,44),T(5,10,5),V(45,55,35)

!IF (TASKID.EQ.0) THEN
IF(I69 .EQ. 9) OPEN(9,FILE='mon',ACCESS='APPEND')
ENDIF

IREIN=0
DO 510 N=1,NCAV
IF(TIMEW(N) .GT. TWAIT) THEN
IREIN=1
TIMEW(N)=0.
NBUB(1)=NBUB(1)+1
IBUB0=NBUB(1)
IF (TASKID.EQ.0) THEN
WRITE(I69,*) 'AddBubble: NBUB(1)=',NBUB(1),ITIME
ENDIF

DO 512 K=K11,KNK
DO 512 J=1,NJ
DO 512 I=1,NI
CC(I,J,K)=F(I,J,K)
SS(I,J,K)=DDF(I,J,K)
IF(I23.EQ.3) FZ1=(ZP(K)-ZCAV(N))**2
512     F(I,J,K)=SQRT((XP(I)-XCAV(N))**2+(YP(J)-Y0)**2+FZ1)-RCV0
CALL DDF_GET

DO 514 K=2,NKM
DO 514 J=2,NJM
DO 514 I=2,NIM
IF(CC(I,J,K).LE.SS(I,J,K) .AND. F(I,J,K).LE.DDF(I,J,K)) THEN
IF (TASKID.EQ.0) THEN
WRITE(I69,*) ' Additional Bubble Merged ',ITIME
ENDIF
NBUB(1)=NBUB(1)-1
IBUB0=IBUB(I,J,K,1)
```

```
GOTO 515
ENDIF
514     CONTINUE
515     CONTINUE

DO 516 K=K11,KNK
DO 516 J=1,NJ
DO 516 I=1,NI
IF(F(I,J,K).LE.DDF(I,J,K)) IBUB(I,J,K,1)=IBUB0
F1=F(I,J,K)/(2.*DDF(I,J,K))
F1=AMAX1(0.,AMIN1(1.,F1))
IF(F(I,J,K).GE.0.) T(I,J,K)=F1*(2.-F1)*T(I,J,K)
F(I,J,K)=AMIN1(CC(I,J,K),F(I,J,K))
516     CONTINUE
CALL DDF_GET
ENDIF
510     CONTINUE

IF(IREIN .EQ. 1) THEN
IREIN=0
CALL IBUB_GET
CALL F_BC
CALL H_GET
CALL T_BC
CALL BVOLB_GET
CALL N_GET
CALL RK_GET
CALL AM_GET
ENDIF
!--------------

!CALL FRAGMENT
!CALL GHBUFFER
!CALL BUFFER
!ALLOCATE(GHVAR(IIST-2:IIEND+2,JJST-2:JJEND+2,KKST-2:KKEND+2,1:2),STAT=IERR)
!IF (IERR /=0) PRINT*,'ERROR IN GHVAR BY',TASKID
!--------------
CALL SOLVE

TIME=TIME+DT
DO 525 N=1,NCAV
TIMEW(N)=TIMEW(N)+DT
I=ICAV(N)
J=JCAV(N)
K=KCAV(N)
IF(F(I,J,K) .LT. DDF(I,J,K)/2.) TIMEW(N)=0.
525     CONTINUE
!*********************************************
UMAX=1.E-10
VMAX=1.E-10
WMAX=1.E-10
PMAX=1.E-10
DO 540 K=K11,KNK
```

```
DO 540 J=1,NJ
DO 540 I=1,NI
UMAX=AMAX1(UMAX,ABS(U(I,J,K)))
VMAX=AMAX1(VMAX,ABS(V(I,J,K)))
WMAX=AMAX1(WMAX,ABS(W(I,J,K)))
540     PMAX=AMAX1(PMAX,ABS(P(I,J,K)))

IF (TASKID.EQ.0) THEN
IF(ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69,4022)&
&     ' ***TIME,DUVWP     =    ',TIME,UMAX/BUMAX,VMAX/BVMAX&
&                                      ,WMAX/BWMAX,PMAX/BPMAX
ENDIF
!-----------------------------------------------
CALL INTP_GET(HTOP1, 2,2,0,2)
CALL INTP_GET(HBOT1,-2,2,0,2)
VTOP=(HTOP1-HTOP)/DT
VBOT=(HBOT1-HBOT)/DT
HTOP=HTOP1
HBOT=HBOT1

!************************************************** OUTPUT
IF(ITIME/JINOUT*JINOUT .EQ. ITIME.OR.ITIME/JI3OUT*JI3OUT .EQ. ITIME) CALL FO_BC
IF(ITIME/JINOUT*JINOUT .EQ. ITIME) CALL INT2D_OUT
IF(ITIME/JI3OUT*JI3OUT .EQ. ITIME) CALL INT3D_OUT
IF(ITIME/JREOUT*JREOUT .EQ. ITIME) CALL RESTF_OUT
IF(ITIME/JFLOUT*JFLOUT .EQ. ITIME) CALL OFILE_OUT(JFLOUT)
IF(ITIME/JPLOUT*JPLOUT .EQ. ITIME) CALL PLOTF_OUT(JPLOUT)

BUMAX=UMAX
BVMAX=VMAX
BWMAX=WMAX
BPMAX=PMAX
IF(I69 .EQ. 9) CLOSE(9)
500     CONTINUE

!------------------------!MPI Functions to deallocate
CALL DEBUFFER
CALL DEGHBUFFER
DEALLOCATE(GHVAR,STAT=IERR)
IF (IERR /=0) PRINT*,'ERROR IN DEALLOCATE GHVAR BY',TASKID
!------------------------

!*********************************************************************
4021    FORMAT(I8,5(1PE12.4))
4022    FORMAT(A,6(1PE12.4))
4023    FORMAT(A,3(I4),4(1PE12.4))
CALL MULTIEND
END
!*****************************************************
!*********************************************************************
SUBROUTINE SOLVE
!*********************************************************************
USE PARAMETERS
```

```
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE CHCFL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE RESLT
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL (KIND=8) :: CF1,UMAX,VMAX,WMAX
!**********************************************************************
CF=1.E10
DO 10 K=2,NKM
DO 10 J=2,NJM
DO 10 I=2,NIM
UMAX=AMAX1(ABS(U(I,J,K)),ABS(U(I,J,K)+AMX(I,J,K)/RHOF(FU(I,J,K))))
VMAX=AMAX1(ABS(V(I,J,K)-VBUB),ABS(V(I,J,K)-VBUB+AMY(I,J,K)/RHOF(FV(I,J,K))))
WMAX=AMAX1(ABS(W(I,J,K)),ABS(W(I,J,K)+AMZ(I,J,K)/RHOF(FW(I,J,K))))
CF1=AMAX1(UMAX/XC(I),VMAX/YC(J),WMAX/ZC(K))
CF1=1./(CF1+1.E-10)
IF(CF1 .GT. CF ) GOTO 10
CF=CF1
ICF=I
JCF=J
KCF=K
10      CONTINUE

DT=AMIN1(DT0,CF*CFO)
IF(DT .LT. 1.E-8) WRITE(I69,*) ' DT IS TOO SMALL'
IF(DT .LT. 1.E-8) STOP
BDEFF=DEFF

DO 110 K=K11,KNK
DO 110 J=1,NJ
DO 110 I=1,NI
110     BF(I,J,K)=F(I,J,K)

DO 120 K=2,NKM
DO 120 J=2,NJM
DO 120 I=2,NIM
120     T(I,J,K)=AMAX1(0.,AMIN1(1.,T(I,J,K)))
CALL T_BC
```

```
CALL UL_GET
CALL TL_GET
IF(NBUB(1) .GE. 1) THEN
CALL IPBUB_GET
CALL BVOLB_MOD
CALL F_GET
CALL F_MOD
IF(JMOVE.EQ.1) CALL F_JMOVE
CALL IBUB_MOD
CALL H_GET
CALL N_GET
CALL RK_GET
ENDIF

CALL T_SOLV
CALL AM_GET
CALL U_SOLV
CALL V_SOLV
CALL W_SOLV
CALL DP_ADD
CALL P_SOLV
CALL PROJT

CALL U_BC
CALL V_BC
CALL W_BC

RETURN
END

!***********************************************************************
SUBROUTINE AM_GET
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE QINTB
USE ROGAM
USE QMSOR
USE FNXYZ
USE RESLT
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
```

```
REAL (KIND=8) :: DUR,DVR,DWR
!***********************************************************************
DO 100 K=K11,KNK
DO 100 J=1,NJ
DO 100 I=1,NI
AM(I,J,K)=0.
AMX(I,J,K)=0.
AMY(I,J,K)=0.
AMZ(I,J,K)=0.
100     VM(I,J,K)=0.

IF(IBOIL.EQ.0) RETURN
CALL QN_GET
DO 200 K=K11,KNK
DO 200 J=1,NJ
DO 200 I=1,NI
AM(I,J,K)=HFGI*(QN(I,J,K,2)/TK2-QN(I,J,K,1)/TK1)
200     CONTINUE
!
DO 300 K=K11,KNK
DO 300 J=1,NJ
DO 300 I=1,NI
IF (I.NE.1)   AMX(I,J,K)=AM(I    ,J,K)*FNX(I    ,J,K)*XDW(I)+AM(I-1,J,K)*FNX(I-1,J,K)*XDE(I)
IF (J.NE.1)   AMY(I,J,K)=AM(I,J    ,K)*FNY(I,J    ,K)*YDS(J)+AM(I,J-1,K)*FNY(I,J-1,K)*YDN(J)
IF(I23.EQ.2) GOTO 300
IF(K.NE.1) AMZ(I,J,K)=AM(I,J,K    )*FNZ(I,J,K    )*ZDB(K)+AM(I,J,K-1)*FNZ(I,J,K-1)*ZDT(K)
300     CONTINUE
!
DO 310 K=2,NKM
DO 310 J=2,NJM
DO 310 I=2,NI
IF(T_1PH(F(I,J,K),F(I-1,J,K))) GOTO 310
IF(S(I,J,K).GT.0. .AND. S(I-1,J,K).GT.0.) GOTO 312
IF(S(I,J,K).LE.0. .AND. S(I-1,J,K).LE.0.) GOTO 310
IF(S(I,J,K)*S(I,J,K)/(S(I,J,K)-S(I-1,J,K)) .LT.S(I,J,K)*F(I,J,K)/(F(I,J,K)-F(I-1,J,K))) GOTO 310
312     DUR=AMX(I,J,K)*VFG*RU(I)*YC(J)*ZC(K)*SIGN1(F(I,J,K)-F(I-1,J,K))
IF(T_2PH(FU(I,J,K),F(I    ,J,K))) VM(I    ,J,K)=VM(I    ,J,K)+DUR
IF(T_2PH(FU(I,J,K),F(I-1,J,K))) VM(I-1,J,K)=VM(I-1,J,K)+DUR
310     CONTINUE
!
DO 320 K=2,NKM
DO 320 J=2,NJ
DO 320 I=2,NIM
IF(T_1PH(F(I,J,K),F(I,J-1,K))) GOTO 320
IF(S(I,J,K).GT.0. .AND. S(I,J-1,K).GT.0.) GOTO 322
IF(S(I,J,K).LE.0. .AND. S(I,J-1,K).LE.0.) GOTO 320
IF(S(I,J,K)*S(I,J,K)/(S(I,J,K)-S(I,J-1,K)) .LT.S(I,J,K)*F(I,J,K)/(F(I,J,K)-F(I,J-1,K))) GOTO 320
322     DVR=AMY(I,J,K)*VFG*RP(I)*XC(I)*ZC(K)*SIGN1(F(I,J,K)-F(I,J-1,K))
IF(T_2PH(FV(I,J,K),F(I,J    ,K))) VM(I,J    ,K)=VM(I,J    ,K)+DVR
IF(T_2PH(FV(I,J,K),F(I,J-1,K))) VM(I,J-1,K)=VM(I,J-1,K)+DVR
320     CONTINUE
!
IF(I23.EQ.2) GOTO 340
```

```
DO 330 K=2,NK
DO 330 J=2,NJM
  DO 330 I=2,NIM
  IF(T_1PH(F(I,J,K),F(I,J,K-1))) GOTO 330
  IF(S(I,J,K).GT.0. .AND. S(I,J,K-1).GT.0.) GOTO 332
  IF(S(I,J,K).LE.0. .AND. S(I,J,K-1).LE.0.) GOTO 330
IF(S(I,J,K)*S(I,J,K)/(S(I,J,K)-S(I,J,K-1)) .LT.S(I,J,K)*F(I,J,K)/(F(I,J,K)-F(I,J,K-1))) GOTO 330
332     DWR=AMZ(I,J,K)*VFG*XC(I)*YC(J)*SIGN1(F(I,J,K)-F(I,J,K-1))
IF(T_2PH(FW(I,J,K),F(I,J,K    ))) VM(I,J,K    )=VM(I,J,K    )+DWR
IF(T_2PH(FW(I,J,K),F(I,J,K-1))) VM(I,J,K-1)=VM(I,J,K-1)+DWR
330     CONTINUE
!
340     IF(QMIC0.LT.1.E-10 .OR. IQ_LIQ.EQ.0) RETURN
CALL QM_GET
RETURN
END
!***********************************************************************
SUBROUTINE BC_ASSS
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE INDMO
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE UVTBC
USE LVFBC
USE LEVFN
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE F2F00
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
LOGICAL T_2PH,T_1PH
REAL (KIND=8) :: DF,HX,HY,HZ,HXE,HXW,HZB,HZT,TTN
INTEGER :: I,J,K,NT,II,JJ,KK
INTEGER :: ITMP,ITMPE,JTMP,JTMPE,KTMP,KTMPE
!***********************************************************************
ENTRY U_BC
!***********************************************************************
CALL SHRINK
IF(IBODY.EQ.0) GOTO 105
DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=3,NIM
```

```
100     IF(SU(I,J,K) .LE. 0.) U(I,J,K)=0.

105     DO 110 K=KKST,KKEND
DO 110 J=JJST,JJEND
IF(IBCW.NE.IOUT) U( 2,J,K)=0.
IF(IBCE.NE.IOUT) U(NI,J,K)=0.
IF(IBCW.EQ.IINF) U( 2,J,K)=UIN0
IF(IBCE.EQ.IINF) U(NI,J,K)=UIN0
110     CONTINUE

ITMPE = IIEND
IF (PIDX.EQ.PROCSX-1) ITMPE = IIEND+1

DO 120 K=KKST,KKEND
DO 120 I=IIST,ITMPE
U(I, 1,K)=0.
U(I,NJ,K)=0.
IF(IBCS.EQ.IOUT .OR. IBCS.EQ.ISYM) U(I, 1,K)=U(I,    2,K)
IF(IBCN.EQ.IOUT .OR. IBCN.EQ.ISYM) U(I,NJ,K)=U(I,NJM,K)
120     CONTINUE


JTMP=JJST
JTMPE=JJEND
ITMPE=IIEND
IF (PIDX.EQ.PROCSX-1) ITMPE = IIEND+1
IF (PIDY.EQ.0) JTMP=JJST-1
IF (PIDY.EQ.PROCSY-1) JTMPE=JJEND+1
IF(I23.EQ.2) RETURN
DO 130 J=JTMP,JTMPE !1,NJ
DO 130 I=IIST,ITMPE !2,NI
U(I,J, 1)=0.
U(I,J,NK)=0.
IF(IBCB.EQ.IOUT .OR. IBCB.EQ.ISYM) U(I,J, 1)=U(I,J,    2)
IF(IBCT.EQ.IOUT .OR. IBCT.EQ.ISYM) U(I,J,NK)=U(I,J,NKM)
130     CONTINUE

CALL FRAGMENT
CALL RELOAD

IF (TASKID.NE.0) THEN
CALL MPI_SEND(U(IIST:IIEND,JJST:JJEND,KKST:KKEND), SPANX*SPANY*SPANZ,
      MPI_DOUBLE_PRECISION, 0, TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL MPI_RECV(U(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),
      RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT
      ,MPI_COMM_WORLD, STATUS,IERR)
ENDDO
ENDIF
```

```
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(U(1,1,1),SIZE(U),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
!***********************************************************************
ENTRY U_BCAS
!***********************************************************************
IF(IBODY.EQ.0) RETURN
DO 140 K=2,NKM
DO 140 J=2,NJM
DO 140 I=3,NIM
IF(SU(I,J,K) .GT. 0.) GOTO 140
AP(I,J,K)=1.
AW(I,J,K)=0.
AE(I,J,K)=0.
AS(I,J,K)=0.
AN(I,J,K)=0.
AB(I,J,K)=0.
AT(I,J,K)=0.
140     CONTINUE
RETURN
!***********************************************************************
ENTRY U_BCSS
!***********************************************************************
DO 150 K=KKST,KKEND
DO 150 J=JJST,JJEND
DO 150 I=IIST,IIEND !3,NIM
150     IF(SU(I,J,K) .LE. 0.) SS(I,J,K)=0.
RETURN
!***********************************************************************
ENTRY UC_SET
!***********************************************************************
DO 160 K=KKST,KKEND
DO 160 J=JJST,JJEND
DO 160 I=IIST,IIEND
160     IF(SU(I,J,K) .LE. 0.) CC(I,J,K)=0.

  IF (TASKID.NE.0) THEN
CALL MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,
      MPI_DOUBLE_PRECISION,0,8,MPI_COMM_WORLD,IERR)
ENDIF
!
IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT)
      ,RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,8
      ,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
!***********************************************************************
```

```
ENTRY V_BC
!************************************************************************
IF(IBODY.EQ.0) GOTO 205
DO 200 K=2,NKM
DO 200 J=3,NJM
DO 200 I=2,NIM
200     IF(SV(I,J,K) .LE. 0.) V(I,J,K)=0.

205     DO 210 K=2,NKM
DO 210 J=3,NJM
V( 1,J,K)=0.
V(NI,J,K)=0.
IF(IBCW.EQ.IOUT .OR. IBCW.EQ.ISYM) V( 1,J,K)=V(    2,J,K)
IF(IBCE.EQ.IOUT .OR. IBCE.EQ.ISYM) V(NI,J,K)=V(NIM,J,K)
210     CONTINUE

DO 220 K=2,NKM
DO 220 I=1,NI
IF(IBCS.NE.IOUT) V(I, 2,K)=0.
IF(IBCN.NE.IOUT) V(I,NJ,K)=0.
IF(IBCS.EQ.IINF) V(I, 2,K)=UIN0
IF(IBCN.EQ.IINF) V(I,NJ,K)=UIN0
IF(IBCN.EQ.IOUT) V(I,NJ,K)=V(I,NJM,K)
220     CONTINUE

IF(I23.EQ.2) RETURN
DO 230 J=2,NJ
DO 230 I=1,NI
IF(IBCB.EQ.IOUT .OR. IBCB.EQ.ISYM) V(I,J, 1)=V(I,J,    2)
IF(IBCT.EQ.IOUT .OR. IBCT.EQ.ISYM) V(I,J,NK)=V(I,J,NKM)
230     CONTINUE
RETURN


!************************************************************************
ENTRY V_BCAS
!************************************************************************
DO 240 K=2,NKM
DO 240 J=3,NJM
DO 240 I=2,NIM
IF(SV(I,J,K) .GT. 0.) GOTO 240
AP(I,J,K)=1.
AW(I,J,K)=0.
AE(I,J,K)=0.
AS(I,J,K)=0.
AN(I,J,K)=0.
AB(I,J,K)=0.
AT(I,J,K)=0.
240     CONTINUE
RETURN
!************************************************************************
ENTRY V_BCSS
!************************************************************************
```

```
DO 250 K=KKST,KKEND
DO 250 J=JJST,JJEND
DO 250 I=IIST,IIEND
250     IF(SV(I,J,K) .LE. 0.) SS(I,J,K)=0.
RETURN
!***********************************************************************
ENTRY VC_SET
!***********************************************************************
DO 260 K=KKST,KKEND
DO 260 J=JJST,JJEND
DO 260 I=IIST,IIEND
260     IF(SV(I,J,K) .LE. 0.) CC(I,J,K)=0.

IF (TASKID.NE.0) THEN
CALL MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,
      MPI_DOUBLE_PRECISION,0,8,MPI_COMM_WORLD,IERR)
ENDIF
!
IF (TASKID.EQ.0) THEN
DO NT=1,26

CALL MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),
      RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,8
      ,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
!***********************************************************************
ENTRY W_BC
!***********************************************************************
IF(I23.EQ.2) RETURN
IF(IBODY.EQ.0) GOTO 305
DO 300 K=3,NKM
DO 300 J=2,NJM
DO 300 I=2,NIM
300     IF(SW(I,J,K) .LE. 0.) W(I,J,K)=0.

305     DO 310 K=3,NKM
DO 310 J=2,NJM
W( 1,J,K)=0.
W(NI,J,K)=0.
IF(IBCW.EQ.IOUT .OR. IBCW.EQ.ISYM) W( 1,J,K)=V(    2,J,K)
IF(IBCE.EQ.IOUT .OR. IBCE.EQ.ISYM) W(NI,J,K)=V(NIM,J,K)
310     CONTINUE

DO 320 K=3,NKM
DO 320 I=1,NI
W(I, 1,K)=0.
W(I,NJ,K)=0.
IF(IBCS.EQ.IOUT .OR. IBCS.EQ.ISYM) W(I, 1,K)=W(I,    2,K)
```

```
IF(IBCN.EQ.IOUT .OR. IBCN.EQ.ISYM) W(I,NJ,K)=W(I,NJM,K)
320     CONTINUE

DO 330 J=1,NJ
DO 330 I=1,NI
IF(IBCB.NE.IOUT) W(I,J, 2)=0.
IF(IBCT.NE.IOUT) W(I,J,NK)=0.
IF(IBCB.EQ.IINF) W(I,J, 2)=UIN0
IF(IBCT.EQ.IINF) W(I,J,NK)=UIN0
330     CONTINUE
RETURN
!***********************************************************************
ENTRY W_BCAS
!***********************************************************************
IF(IBODY.EQ.0) RETURN
DO 340 K=3,NKM
DO 340 J=2,NJM
DO 340 I=2,NIM
IF(SW(I,J,K) .GT. 0.) GOTO 340
AP(I,J,K)=1.
AW(I,J,K)=0.
AE(I,J,K)=0.
AS(I,J,K)=0.
AN(I,J,K)=0.
AB(I,J,K)=0.
AT(I,J,K)=0.
340     CONTINUE
RETURN
!***********************************************************************
ENTRY W_BCSS
!***********************************************************************
DO 350 K=KKST,KKEND
DO 350 J=JJST,JJEND
DO 350 I=IIST,IIEND
350     IF(SW(I,J,K) .LE. 0.) SS(I,J,K)=0.
RETURN
!***********************************************************************
ENTRY WC_SET
!***********************************************************************
DO 360 K=KKST,KKEND
DO 360 J=JJST,JJEND
DO 360 I=IIST,IIEND
360     IF(SW(I,J,K) .LE. 0.) CC(I,J,K)=0.

  IF (TASKID.NE.0) THEN
CALL MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,
      MPI_DOUBLE_PRECISION,0,8,MPI_COMM_WORLD,IERR)
ENDIF
!
IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),
      RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,8,
```

```
      MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
!************************************************************************
ENTRY T_BC
!************************************************************************
DO 400 K=2,NKM
DO 400 J=2,NJM
DO 400 I=2,NIM
IF(IQ_VAP.EQ.0 .AND. F(I,J,K).LT.0.) T(I,J,K)=TINT
IF(IQ_LIQ.EQ.0 .AND. F(I,J,K).GT.0.) T(I,J,K)=TINT
IF(F(I,J,K) .EQ. 0.) T(I,J,K)=TINT
IF(S(I,J,K) .LE. 0.) T(I,J,K)=TWAL
400     CONTINUE

DO 410 K=2,NKM
DO 410 J=2,NJM
T( 1,J,K)=T(   2,J,K)
410     T(NI,J,K)=T(NIM,J,K)

DO 420 K=2,NKM
DO 420 I=1,NI
T(I, 1,K)=T(I,   2,K)
420     T(I,NJ,K)=T(I,NJM,K)

IF(I23.EQ.3) THEN
DO 430 J=1,NJ
DO 430 I=1,NI
T(I,J, 1)=T(I,J,   2)
430     T(I,J,NK)=T(I,J,NKM)
ENDIF

IF(IBCS.EQ.IWAL) THEN
DO 422 K=2,NKM
DO 422 I=2,NIM
HY=1.
TTN=T(I,2,K)
IF(F(I,1,K).LE.0. .AND. IQ_VAP.EQ.0) HY=0.
IF(F(I,1,K).GE.0. .AND. IQ_LIQ.EQ.0) HY=0.
IF(T_2PH(F(I,1,K),F(I,2,K))) THEN
HY=F(I,1,K)/(F(I,1,K)-F(I,2,K))
TTN=TINT
ENDIF
HXW=0.5
HXE=0.5
IF(T_2PH(F(I,1,K),F(I-1,1,K))) HXW=XD(I   )*F(I,1,K)/(F(I,1,K)-F(I-1,1,K))/XC(I)
IF(T_2PH(F(I,1,K),F(I+1,1,K))) HXE=XD(I+1)*F(I,1,K)/(F(I,1,K)-F(I+1,1,K))/XC(I)
HX=HXW+HXE
HZ=1.
```

```
IF(I23.EQ.3) THEN
HZB=0.5
HZT=0.5
IF(T_2PH(F(I,1,K),F(I,1,K-1))) HZB=ZD(K    )*F(I,1,K)/(F(I,1,K)-F(I,1,K-1))/ZC(K)
IF(T_2PH(F(I,1,K),F(I,1,K+1))) HZT=ZD(K+1)*F(I,1,K)/(F(I,1,K)-F(I,1,K+1))/ZC(K)
HZ=HZB+HZT
ENDIF
T(I, 1,K)=TTN+(TBCS-TTN)*HY*HX*HZ
422     CONTINUE
ENDIF
RETURN
!************************************************************************
ENTRY T_BCAS
!************************************************************************
DO 440 K=2,NKM
DO 440 J=2,NJM
DO 440 I=2,NIM
IF( (IQ_VAP.EQ.0 .AND. F(I,J,K).LT.0.) .OR. (IQ_LIQ.EQ.0 .AND. F(I,J,K).GT.0.) .OR.&
      &   (S(I,J,K).LE.0. .OR. F(I,J,K).EQ.0.)) THEN
AP(I,J,K)=1.
AW(I,J,K)=0.
AE(I,J,K)=0.
AS(I,J,K)=0.
AN(I,J,K)=0.
AB(I,J,K)=0.
AT(I,J,K)=0.
ENDIF
440     CONTINUE
RETURN
!************************************************************************
ENTRY T_BCSS
!************************************************************************
DO 450 K=KKST,KKEND
DO 450 J=JJST,JJEND
DO 450 I=IIST,IIEND
450     IF( (IQ_VAP.EQ.0 .AND. F(I,J,K).LT.0.).OR.(IQ_LIQ.EQ.0 .AND. F(I,J,K).GT.0.).OR.(S(I,J,K).LE.0.
      .OR. F(I,J,K).EQ.0.)) SS(I,J,K)=0.
RETURN
!************************************************************************
ENTRY TC_SET
!************************************************************************
DO 460 K= KKST,KKEND
DO 460 J= JJST,JJEND
DO 460 I= IIST,IIEND
460     IF( (IQ_VAP.EQ.0 .AND. F(I,J,K).LT.0.) .OR. (IQ_LIQ.EQ.0 .AND. F(I,J,K).GT.0.) .OR.
      (S(I,J,K).LE.0. .OR. F(I,J,K).EQ.0.)) CC(I,J,K)=0.
IF (TASKID.NE.0) THEN
CALL MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,
      MPI_DOUBLE_PRECISION,0,8,MPI_COMM_WORLD,IERR)
ENDIF
!
IF (TASKID.EQ.0) THEN
DO NT=1,26
```

```
CALL MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),
     RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,8,
     MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
!***********************************************************************
ENTRY P_BC
!***********************************************************************
CALL SHRINK
ITMP=IIST
JTMP=JJST
KTMP=KKST
ITMPE=IIEND
JTMPE=JJEND
KTMPE=KKEND

DO 510 K=KKST,KKEND !2,NKM
DO 510 J=JJST,JJEND !2,NJM
P( 1,J,K)=P(    2,J,K)
P(NI,J,K)=P(NIM,J,K)
IF(IBCW.EQ.IOUT) P( 1,J,K)=0.
IF(IBCE.EQ.IOUT) P(NI,J,K)=0.
510     CONTINUE

IF (PIDX.EQ.0) ITMP=1
IF (PIDX.EQ.PROCSX-1) ITMPE=ID
DO 520 K=KKST,KKEND
DO 520 I=ITMP,ITMPE
P(I, 1,K)=P(I,    2,K)
P(I,NJ,K)=P(I,NJM,K)
IF(IBCS.EQ.IOUT) P(I, 1,K)=0.
IF(IBCN.EQ.IOUT) P(I,NJ,K)=0.
520     CONTINUE

IF (PIDX.EQ.0) ITMP=1
IF (PIDX.EQ.PROCSX-1) ITMPE=ID
IF (PIDY.EQ.0) JTMP=1
IF (PIDY.EQ.PROCSY-1) JTMPE=JD
IF(I23.EQ.2) RETURN
DO 530 J=JTMP,JTMPE
DO 530 I=ITMP,ITMPE
P(I,J, 1)=P(I,J,    2)
P(I,J,NK)=P(I,J,NKM)
IF(IBCB.EQ.IOUT) P(I,J, 1)=0.
IF(IBCT.EQ.IOUT) P(I,J,NK)=0.
530     CONTINUE

CALL FRAGMENT
CALL RELOAD
IF (TASKID.NE.0) THEN
```

```
CALL MPI_SEND(P(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,
     MPI_DOUBLE_PRECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL MPI_RECV(P(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),
     RSPANZ(NT):RSPANZE(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,
     MPI_COMM_WORLD,STATUS,IERR)
ENDD
ENDIF
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(P(1,1,1),SIZE(P),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
CALL SHRINK
CALL RELOAD
RETURN
!***********************************************************************
ENTRY F_BC
!***********************************************************************
IF(IBODY.EQ.1 .AND. ICONB.EQ.0) THEN
DO 600 K=2,NKM
DO 600 J=2,NJM
DO 600 I=2,NIM
IF(S(I,J,K).LE.0.) F(I,J,K)=SIGN(F(I,J,K),BF(I,J,K))
600     CONTINUE
ENDIF

DO 610 K=2,NKM
DO 610 J=2,NJM
F( 1,J,K)=F(   2,J,K)
F(NI,J,K)=F(NIM,J,K)
610     CONTINUE

DO 620 K=2,NKM
DO 620 I=1,NI
F(I, 1,K)=F(I,   2,K)
620     F(I,NJ,K)=F(I,NJM,K)

IF(I23.EQ.3) THEN
DO 630 J=1,NJ
DO 630 I=1,NI
F(I,J, 1)=F(I,J,   2)
F(I,J,NK)=F(I,J,NKM)
630     CONTINUE
ENDIF

IF(IBCS.GE.IWAL) THEN
DO 622 K=2,NKM
DO 622 I=2,NIM
IF(ICONT.EQ.1) THEN
DF=(BF(I,1,K)-F(I,2,K))/YD(2)
DF=AMIN1(AMAX1(DF,COSMIN),COSMAX)
COSBS(I,K)=DF
```

```
F(I,1,K)=F(I,2,K)+DF*YD(2)
ELSE
F(I,1,K)=F(I,2,K)-XDW(3)*(F(I,3,K)-F(I,2,K))
ENDIF
IF(ICONT.EQ.0) F(I,1,K)=SIGN(F(I,1,K),BF(I,1,K))
622     CONTINUE
ENDIF
RETURN
!***********************************************************************
ENTRY IBUB_BC
!***********************************************************************
DO 1700 I2B=1,N2B
DO 710 K=2,NKM
DO 710 J=2,NJM
IBUB( 1,J,K,I2B)=IBUB(    2,J,K,I2B)
710     IBUB(NI,J,K,I2B)=IBUB(NIM,J,K,I2B)

DO 720 K=2,NKM
DO 720 I=1,NI
IBUB(I, 1,K,I2B)=IBUB(I,    2,K,I2B)
720     IBUB(I,NJ,K,I2B)=IBUB(I,NJM,K,I2B)

IF(I23.EQ.2) RETURN
DO 730 J=1,NJ
DO 730 I=1,NI
IBUB(I,J, 1,I2B)=IBUB(I,J,    2,I2B)
730     IBUB(I,J,NK,I2B)=IBUB(I,J,NKM,I2B)
1700    CONTINUE
RETURN
!***********************************************************************
ENTRY FO_BC
!***********************************************************************
DO 900 K=K11,KNK
DO 900 J=1,NJ
DO 900 I=1,NI
900     FO(I,J,K)=F(I,J,K)

DO 910 K=1,NK
DO 910 J=1,NJ
IF(IBCW.EQ.ISYM) FO(0,J,K)=FO(1,J,K)
910     CONTINUE

IF(I23.EQ.2) RETURN
DO 920 K=1,NK
DO 920 I=0,NI+1
IF(IBCS.EQ.ISYM) FO(I,0,K)=FO(I,1,K)
920     CONTINUE

IF(I23.EQ.2) RETURN
DO 930 J=0,NJ+1
DO 930 I=0,NI+1
IF(IBCB.EQ.ISYM) FO(I,J,0)=FO(I,J,1)
930     CONTINUE
```

```
RETURN
END
!***********************************************************************
SUBROUTINE BUB_SUB
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE FFSOR
USE QMSOR
USE RESLT
USE IBSUB
USE ICHEK
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,II,JJ,KK,IND0,IND1,IND2,IND9,N,NBUB1,IBUB0,IBUB1,IBUBS,IFX,IFY,IFZ
INTEGER (KIND=4), DIMENSION(100) :: MKBUB,IPBUBS
REAL (KIND=8) :: AFP,FFP,V00,V11,BVOLBS,VOLHB
!***********************************************************************
ENTRY BUB_BRK
!***********************************************************************
IF(NBUB(I2B) .EQ.     0) RETURN
IF(NBUB(I2B) .EQ. 100) RETURN
IF(ITIME .NE. ITIME/10*10) RETURN
V00=V0H(I2B)
V11=V1H(I2B)

DO 10 N=1,NBUB(I2B)
10       MKBUB(N)=0

DO 20 K=2,NKM
DO 20 J=2,NJM
DO 20 I=2,NIM
20       MK(I,J,K)=0
!---------------------------------
100     CONTINUE
NCHB=0
DO 110 K=2,NKM
DO 110 J=2,NJM
DO 110 I=2,NIM
IF(    MK(I,J,K)        .EQ. 1) GOTO 110
IF(IBUB(I,J,K,I2B) .EQ. 0) GOTO 110
NCHB=1
IIB(1)=I
JJB(1)=J
```

```
KKB(1)=K
MK(I,J,K)=1
IBUB0=IBUB(I,J,K,I2B)
GOTO 115
110     CONTINUE
115     IF(NCHB.EQ.0) GOTO 190

N=0
120     N=N+1
IF(N .GT. NCHB) GOTO 135
II=IIB(N)
JJ=JJB(N)
KK=KKB(N)
DO 130 K=KK-1,KK+1
DO 130 J=JJ-1,JJ+1
DO 130 I=II-1,II+1
IF(    MK(I,J,K)         .EQ.         1) GOTO 130
IF(IBUB(I,J,K,I2B) .NE. IBUB0) GOTO 130
MK(I,J,K)=1
NCHB=NCHB+1
IIB(NCHB)=I
JJB(NCHB)=J
KKB(NCHB)=K
130     CONTINUE
GOTO 120
135     CONTINUE

IF(MKBUB(IBUB0).EQ.0) THEN
MKBUB(IBUB0)=IBUB0
GOTO 100
ENDIF
IBUB0=MKBUB(IBUB0)

WRITE(I69,9000) ' Bubble Area Breaking !!!'&
&                       ,ITIME,IBUB0,IIB(1),JJB(1),KKB(1)
9000    FORMAT(A,I8,I4,3(I4))
NBUB(I2B)=NBUB(I2B)+1
NBUB1=NBUB(I2B)
BVOLB(NBUB1,I2B)=0.
DO 150 N=1,NCHB
IBUB(IIB(N),JJB(N),KKB(N),I2B)=NBUB1
VOLHB=VOL(IIB(N),JJB(N),KKB(N))*HB(IIB(N),JJB(N),KKB(N))
BVOLB(NBUB1,I2B)=BVOLB(NBUB1,I2B)&
&                       +(V00-V11*H(IIB(N),JJB(N),KKB(N)))*VOLHB
150     CONTINUE
BVOLB(IBUB0,I2B)=BVOLB(IBUB0,I2B)-BVOLB(NBUB1,I2B)
IF(BVOLB(NBUB1,I2B).GT.BVOLB(IBUB0,I2B)) THEN
BVOLBS=BVOLB(IBUB0,I2B)
BVOLB(IBUB0,I2B)=BVOLB(NBUB1,I2B)
BVOLB(NBUB1,I2B)=BVOLBS
DO 160 K=2,NKM
DO 160 J=2,NJM
DO 160 I=2,NIM
```

```
IBUBS=IBUB(I,J,K,I2B)
IF(IBUBS.EQ.IBUB0) IBUB(I,J,K,I2B)=NBUB1
160     IF(IBUBS.EQ.NBUB1) IBUB(I,J,K,I2B)=IBUB0
ENDIF
IPBUB(NBUB1,I2B)=IPBUB(IBUB0,I2B)
MKBUB(NBUB1)=NBUB1
WRITE(I69,9001) ' NBUB=',NBUB(I2B),' I2B=',I2B,' NCHB=',NCHB,&
&                        ' IBUB0=',IBUB0&
&                       ,' VOLB=',BVOLB(NBUB(I2B),I2B),BVOLB(IBUB0,I2B)
9001    FORMAT(A,I4,A,I4,A,I5,A,I4,A,1PE12.4,1PE12.4)
GOTO 100
190     CONTINUE

RETURN
!***********************************************************************
ENTRY BUB_MRG(IND1,IND2)
!***********************************************************************
IND0=MIN(IND1,IND2)
IND9=MAX(IND1,IND2)

BVOLB(IND0,I2B)=BVOLB(IND0,I2B)+BVOLB(IND9,I2B)

DO 300 K=2,NKM
DO 300 J=2,NJM
DO 300 I=2,NIM
IF(IBUB(I,J,K,I2B) .EQ. IND9) IBUB(I,J,K,I2B)=IND0
300     IF(IBUB(I,J,K,I2B) .GT. IND9) IBUB(I,J,K,I2B)=IBUB(I,J,K,I2B)-1

DO 310 N=IND9,NBUB(I2B)-1
IPBUB(N,I2B)=IPBUB(N+1,I2B)
310     BVOLB(N,I2B)=BVOLB(N+1,I2B)
BVOLB(NBUB(I2B),I2B)=0.
NBUB(I2B)=NBUB(I2B)-1

RETURN
!***********************************************************************
ENTRY IPBUB_GET
!***********************************************************************
IF(N2B.EQ.1) RETURN
DO 400 N=1,NBUB(2)
IPBUBS(N)=IPBUB(N,2)
400     IPBUB(N,2)=0

IF(NBUB(2).EQ.1) RETURN
J=2
DO 410 K=2,NKM
DO 410 I=2,NIM
IBUB1=IBUB(I,J,K,2)
IF(IBUB1.LE.1) GOTO 410
IPBUB(IBUB1,2)=1
410     CONTINUE

DO 420 N=1,NBUB(2)
```

```
IF(IPBUBS(N).EQ.IPBUB(N,2)) GOTO 420
WRITE(I69,*) 'Bubble: IPBUB=',IPBUB(N,2),N
420     CONTINUE

RETURN
!***********************************************************************
ENTRY IFFIX_GET
!***********************************************************************
DO 500 K=2,NKM
DO 500 J=2,NJM
DO 500 I=2,NIM
500     IFFIX(I,J,K)=0

DO 510 K=2,NKM
DO 510 J=2,NJM
DO 510 I=2,NIM
FFP=ABS(F(I,J,K))
IF(FFP.GE.ABS(F(I-1,J,K))) GOTO 510
IF(FFP.GE.ABS(F(I+1,J,K))) GOTO 510
IF(FFP.GE.ABS(F(I,J-1,K))) GOTO 510
IF(FFP.GE.ABS(F(I,J+1,K))) GOTO 510
IF(I23.EQ.3) THEN
IF(FFP.GE.ABS(F(I,J,K-1))) GOTO 510
IF(FFP.GE.ABS(F(I,J,K+1))) GOTO 510
ENDIF
IFFIX(I,J,K)=1
510     CONTINUE

DO 520 K=2,NKM
DO 520 J=2,NJM
DO 520 I=2,NIM
IF(IFFIX(I,J,K).EQ.1) GOTO 520
IFX=MAX(IFFIX(I-1,J,K),IFFIX(I+1,J,K))
IFY=MAX(IFFIX(I,J-1,K),IFFIX(I,J+1,K))
FFP=F(I,J,K)
AFP=ABS(FFP)
IF(IFX.EQ.1) GOTO 524
IF(FFP*F(I-1,J,K).LT.0..AND.AFP.LT.ABS(F(I-1,J,K))) IFFIX(I,J,K)=1
IF(FFP*F(I+1,J,K).LT.0..AND.AFP.LT.ABS(F(I+1,J,K))) IFFIX(I,J,K)=1
524     IF(IFY.EQ.1) GOTO 526
IF(FFP*F(I,J-1,K).LT.0..AND.AFP.LT.ABS(F(I,J-1,K))) IFFIX(I,J,K)=1
IF(FFP*F(I,J+1,K).LT.0..AND.AFP.LT.ABS(F(I,J+1,K))) IFFIX(I,J,K)=1
526     IF(I23.EQ.2) GOTO 520
IFZ=MAX(IFFIX(I,J,K-1),IFFIX(I,J,K+1))
IF(IFZ.EQ.1) GOTO 520
IF(FFP*F(I,J,K-1).LT.0..AND.AFP.LT.ABS(F(I,J,K-1))) IFFIX(I,J,K)=1
IF(FFP*F(I,J,K+1).LT.0..AND.AFP.LT.ABS(F(I,J,K+1))) IFFIX(I,J,K)=1
520     CONTINUE

RETURN
END

!***********************************************************************
```

```
SUBROUTINE BUBBLES
!***********************************************************************

USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE IDREF
USE LEVFN
USE FFSOR
USE QMSOR
USE RESLT
USE IBSUB
USE ICHEK
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,II,JJ,KK,N1,N2,I1,I2,N,NN,IND1,IBUB0,IBUB1,IIMAX,NIMAX,NBUB0,NIBUB(100)
REAL*8 :: DVOLB(100,2),FLUX,FMIN,ROLIQ,ROVAP,V00,V11,VOLH
!***********************************************************************
ENTRY BVOLB_GET
!***********************************************************************
DO 1000 I2B=1,N2B
V00=V0H(I2B)
V11=V1H(I2B)

DO 100 N=1,NBUB(I2B)
100     BVOLB(N,I2B)=0.

DO 110 K=2,NKM
DO 110 J=2,NJM
DO 110 I=2,NIM
N=IBUB(I,J,K,I2B)
IF(N.LT.1) GOTO 110
VOLH=(V00-V11*H(I,J,K))*VOL(I,J,K)*HB(I,J,K)
BVOLB(N,I2B)=BVOLB(N,I2B)+VOLH
110     CONTINUE
1000    CONTINUE

RETURN
!***********************************************************************
ENTRY BVOLB_MOD
!***********************************************************************
DO 200 I2B=1,N2B
DO 200 N=1,NBUB(I2B)
200     DVOLB(N,I2B)=0.

IF(IBOIL.EQ.1) THEN
ROLIQ=-RO1/(RO2-RO1)
```

```
ROVAP= RO2/(RO2-RO1)
DO 202 K=2,NKM
DO 202 J=2,NJM
DO 202 I=2,NIM
I1=IBUB(I,J,K,1)
I2=IBUB(I,J,K,2)
IF(I1.GE.1) DVOLB(I1,1)=DVOLB(I1,1)+VM(I,J,K)
IF(I2.GE.1) DVOLB(I2,2)=DVOLB(I2,2)+VM(I,J,K)
202     CONTINUE
DO 204 N=1,NBUB(1)
204     DVOLB(N,1)=DVOLB(N,1)*ROVAP
DO 206 N=1,NBUB(2)
206     DVOLB(N,2)=DVOLB(N,2)*ROLIQ
ENDIF

DO 1200 I2B=1,N2B
V00=V0H(I2B)
V11=V1H(I2B)
DO 210 K=2,NKM
DO 210 J=2,NJM
N=IBUB(    2,J,K,I2B)
IF(N.GE.1) THEN
FLUX=U( 2,J,K)*RU( 2)*YC(J)*ZC(K)*(V00-V11*H( 1,J,K))
DVOLB(N,I2B)=DVOLB(N,I2B)+FLUX
ENDIF

N=IBUB(NIM,J,K,I2B)
IF(N.GE.1) THEN
FLUX=U(NI,J,K)*RU(NI)*YC(J)*ZC(K)*(V00-V11*H(NI,J,K))
DVOLB(N,I2B)=DVOLB(N,I2B)-FLUX
ENDIF
210     CONTINUE

DO 220 K=2,NKM
DO 220 I=2,NIM
N=IBUB(I,    2,K,I2B)
IF(N.GE.1) THEN
FLUX=V(I, 2,K)*RP(I)*XC(I)*ZC(K)*(V00-V11*H(I, 1,K))
DVOLB(N,I2B)=DVOLB(N,I2B)+FLUX
ENDIF

N=IBUB(I,NJM,K,I2B)
IF(N.GE.1) THEN
FLUX=V(I,NJ,K)*RP(I)*XC(I)*ZC(K)*(V00-V11*H(I,NJ,K))
DVOLB(N,I2B)=DVOLB(N,I2B)-FLUX
ENDIF
220     CONTINUE

IF(I23.EQ.2) GOTO 235
DO 230 J=2,NJM
DO 230 I=2,NIM
N=IBUB(I,J,    2,I2B)
IF(N.GE.1) THEN
```

```
FLUX=W(I,J, 2)*XC(I)*YC(J)*(V00-V11*H(I,J, 1))
DVOLB(N,I2B)=DVOLB(N,I2B)+FLUX
ENDIF

N=IBUB(I,J,NKM,I2B)
IF(N.GE.1) THEN
FLUX=W(I,J,NK)*XC(I)*YC(J)*(V00-V11*H(I,J,NK))
DVOLB(N,I2B)=DVOLB(N,I2B)-FLUX
ENDIF
230     CONTINUE

235     CONTINUE
DO 240 N=1,NBUB(I2B)
240     BVOLB(N,I2B)=BVOLB(N,I2B)+DVOLB(N,I2B)*DT

1200    CONTINUE

RETURN
!**********************************************************************
ENTRY IBUB_GET
!**********************************************************************
DO 1300 I2B=1,N2B
V11=V1H(I2B)
DO 300 K=2,NKM
DO 300 J=2,NJM
DO 300 I=2,NIM
IF(V11*F(I,J,K) .GT. DDF(I,J,K)) IBUB(I,J,K,I2B)=0
300     IF(HB(I,J,K).LT.1.E-8) IBUB(I,J,K,I2B)=0
!------------------------------------- F<3*DL & IBUB=0
DO 310 K=2,NKM
DO 310 J=2,NJM
DO 310 I=2,NIM
IF(V11*F(I,J,K) .GT. DDF(I,J,K)) GOTO 310
IF(HB(I,J,K).LT.1.E-8) GOTO 310
IF(IBUB(I,J,K,I2B) .NE. 0) GOTO 310
N=0
312     N=N+1
FMIN=1.E30
DO 314 KK=MAX(2,K-N),MIN(NKM,K+N)
DO 314 JJ=MAX(2,J-N),MIN(NJM,J+N)
DO 314 II=MAX(2,I-N),MIN(NIM,I+N)
IF(IBUB(II,JJ,KK,I2B).EQ.0) GOTO 314
IF(ABS(F(I,J,K)-F(II,JJ,KK)).GT.FMIN) GOTO 314
IBUB(I,J,K,I2B)=IBUB(II,JJ,KK,I2B)
FMIN=ABS(F(I,J,K)-F(II,JJ,KK))
314     CONTINUE
IF(IBUB(I,J,K,I2B) .NE. 0) GOTO 310
WRITE(I69,9000) ' Bubble ID=0 for F < FDV ',ITIME,I,J,K
IF(N .LE. 5) GOTO 312
F(I,J,K)=V11*2.*DDF(I,J,K)
IF(IBUB(I,J,K,I2B) .EQ. 0)&
&WRITE(I69,9000) ' Bubble ID=0 for F < FDV while N=',N,I2B
9000    FORMAT(A,I8,4(I4))
```

```
310     CONTINUE
1300    CONTINUE

CALL IBUB_BC

RETURN
!**************************************************************************
ENTRY IBUB_MOD
!**************************************************************************
DO 1400 I2B=1,N2B
V11=V1H(I2B)
IF(NBUB(I2B).EQ.1) GOTO 435
!--------------------------------------- Check Bubble Merging
DO 410 K=2,NKM
DO 410 J=2,NJM
DO 410 I=2,NIMM
N1=IBUB(I   ,J,K,I2B)
N2=IBUB(I+1,J,K,I2B)
IF(N1.LE.0 .OR. N2.LE.0 .OR. N1.EQ.N2) GOTO 410
IF(V11*(F(I,J,K)+F(I+1,J,K)) .GT. &
&       (DDF(I,J,K)+DDF(I+1,J,K))/2) GOTO 410
WRITE(I69,9001) ' Bubble Area Merging:x !!'&
&                       ,ITIME,I2B,N1,N2,I,I+1,J,K
CALL BUB_MRG(N1,N2)
410     CONTINUE
9001    FORMAT(A,I8,8(I4))

DO 420 K=2,NKM
DO 420 J=2,NJMM
DO 420 I=2,NIM
N1=IBUB(I,J   ,K,I2B)
N2=IBUB(I,J+1,K,I2B)
IF(N1.LE.0 .OR. N2.LE.0 .OR. N1.EQ.N2) GOTO 420
IF(V11*(F(I,J,K)+F(I,J+1,K)) .GT. &
&       (DDF(I,J,K)+DDF(I,J+1,K))/2) GOTO 420
WRITE(I69,9001) ' Bubble Area Merging:y !!'&
&                       ,ITIME,I2B,N1,N2,I,J,J+1,K
CALL BUB_MRG(N1,N2)
420     CONTINUE

IF(I23.EQ.2) GOTO 435
DO 430 K=2,NKMM
DO 430 J=2,NJM
DO 430 I=2,NIM
N1=IBUB(I,J,K   ,I2B)
N2=IBUB(I,J,K+1,I2B)
IF(N1.LE.0 .OR. N2.LE.0 .OR. N1.EQ.N2) GOTO 430
IF(V11*(F(I,J,K)+F(I,J,K+1)) .GT. &
&       (DDF(I,J,K)+DDF(I,J,K+1))/2) GOTO 430
WRITE(I69,9001) ' Bubble Area Merging:z !!'&
&                       ,ITIME,I2B,N1,N2,I,J,K,K+1
CALL BUB_MRG(N1,N2)
430     CONTINUE
```

```
!--------------------------------------- Check Bubble Breaking
435     CALL BUB_BRK
1400    CONTINUE
!-----------------------------------------------
DO 1500 I2B=1,N2B
NBUB0=NBUB(I2B)
DO 500 N=1,NBUB(I2B)
500     NIBUB(N)=0

DO 510 K=2,NKM
DO 510 J=2,NJM
DO 510 I=2,NIM
N=IBUB(I,J,K,I2B)
IF(N .LE. 0) GOTO 510
NIBUB(N)=NIBUB(N)+1
510     CONTINUE

N=0
520     N=N+1
IF(N .GT. NBUB(I2B)) GOTO 530
IF(NIBUB(N) .GT. 0) GOTO 520
IBUB0=N
DO 522 NN=N,NBUB(I2B)-1
NIBUB(NN)=NIBUB(NN+1)
522     BVOLB(NN,I2B)=BVOLB(NN+1,I2B)
NIBUB(NBUB(I2B))=0
BVOLB(NBUB(I2B),I2B)=0.
DO 524 K=2,NKM
DO 524 J=2,NJM
DO 524 I=2,NIM
524     IF(IBUB(I,J,K,I2B) .GT. IBUB0) IBUB(I,J,K,I2B)=IBUB(I,J,K,I2B)-1
NBUB(I2B)=NBUB(I2B)-1
N=N-1
GOTO 520

530     CONTINUE
IF(NBUB(I2B) .NE. NBUB0) WRITE(I69,*) &
&    ' MIBubble: NBUB(I2B)=',NBUB(I2B),I2B,ITIME
1500    CONTINUE

CALL IBUB_BC

RETURN
!***********************************************************************
ENTRY IBUB_INIT
!***********************************************************************
!       DO 1600 I2B=1,N2B
NBUB(I2B)=0
V11=V1H(I2B)
DO 590 K=2,NKM
DO 590 J=2,NJM
DO 590 I=2,NIM
MK(I,J,K)=0
```

```
IBUB(I,J,K,I2B)=1
IF(V11*F(I,J,K).GT.DDF(I,J,K)) IBUB(I,J,K,I2B)=0
IF(HB(I,J,K).LT.1.E-8) IBUB(I,J,K,I2B)=0
IF(IBUB(I,J,K,I2B).EQ.1) NBUB(I2B)=1
590     CONTINUE
IF(NBUB(I2B).EQ.0) GOTO 1600
!--------------------------------------- Check Multi bubbles
600     CONTINUE
NCHB=0
DO 610 K=2,NKM
DO 610 J=2,NJM
DO 610 I=2,NIM
IF(    MK(I,J,K)       .EQ.1) GOTO 610
IF(IBUB(I,J,K,I2B).EQ.0) GOTO 610
NCHB=1
IIB(1)=I
JJB(1)=J
KKB(1)=K
MK(I,J,K)=1
IBUB0=IBUB(I,J,K,I2B)
GOTO 615
610     CONTINUE
615     IF(NCHB.EQ.0) GOTO 665

N=0
620     N=N+1
IF(N .GT. NCHB) GOTO 635
II=IIB(N)
JJ=JJB(N)
KK=KKB(N)
DO 630 K=KK-1,KK+1
DO 630 J=JJ-1,JJ+1
DO 630 I=II-1,II+1
IF(    MK(I,J,K)         .EQ.         1) GOTO 630
IF(IBUB(I,J,K,I2B) .NE. IBUB0) GOTO 630
MK(I,J,K)=1
NCHB=NCHB+1
IIB(NCHB)=I
JJB(NCHB)=J
KKB(NCHB)=K
630     CONTINUE
GOTO 620

635     IF(NCHB .LT. 3) THEN
DO 640 K=2,NKM
DO 640 J=2,NJM
DO 640 I=2,NIM
640     IF(IBUB(I,J,K,I2B).EQ.NBUB(I2B)) IBUB(I,J,K,I2B)=NBUB(I2B)-1
NBUB(I2B)=MAX(0,NBUB(I2B)-1)
GOTO 600
ENDIF

WRITE(I69,9002) ' Number of Bubbles=',NBUB(I2B),I2B,NCHB
```

```
NIBUB(NBUB(I2B))=NCHB
9002   FORMAT(A,I8,I4,I8,8(I4))
IND1=0
DO 650 K=2,NKM
DO 650 J=2,NJM
DO 650 I=2,NIM
IF(   MK(I,J,K)        .EQ.        1) GOTO 650
IF(IBUB(I,J,K,I2B) .NE. IBUB0) GOTO 650
IBUB(I,J,K,I2B)=NBUB(I2B)+1
IND1=IND1+1
650    CONTINUE
IF(IND1.EQ.0) GOTO 665
NBUB(I2B)=NBUB(I2B)+1
GOTO 600

665    CONTINUE

IF(NBUB(I2B).LE.1) GOTO 1600
NIMAX=0
DO 670 N=1,NBUB(I2B)
IF(NIBUB(N).LT.NIMAX) GOTO 670
NIMAX=NIBUB(N)
IIMAX=N
670    CONTINUE
IF(IIMAX.EQ.1) GOTO 1600
DO 680 K=2,NKM
DO 680 J=2,NJM
DO 680 I=2,NIM
IBUB1=IBUB(I,J,K,I2B)
IF(IBUB1.EQ.       1) IBUB(I,J,K,I2B)=IIMAX
IF(IBUB1.EQ.IIMAX) IBUB(I,J,K,I2B)=1
680    CONTINUE

1600   CONTINUE

RETURN
END
!***********************************************************************
SUBROUTINE DDF_GET
!***********************************************************************

USE PARAMETERS
USE ICHOS
USE IGRID
USE GRID1
USE GRIDB
USE LEVFN
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL*8 :: FX1,FY1,DX1,DY1,DZ1,DET
!***********************************************************************
IF(IUGRD.EQ.1) RETURN
```

```
CALL FUVW_GET

DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
FX1=(FU(I+1,J,K)-FU(I,J,K))/XC(I)
FY1=(FV(I,J+1,K)-FV(I,J,K))/YC(J)
IF(I23.EQ.3) FZ1=(FW(I,J,K+1)-FW(I,J,K))/ZC(K)
DET=AMAX1(SQRT(FX1**2+FY1**2+FZ1**2),1.E-10)
FX1=FX1/DET
FY1=FY1/DET
FZ1=FZ1/DET
DX1=FX1*XC(I)
DY1=FY1*YC(J)
DZ1=FZ1*ZC(K)
100     DDF(I,J,K)=AMAX1(DTF(I,J,K),SQRT(DX1**2+DY1**2+DZ1**2))

DO 110 K=2,NKM
DO 110 J=2,NJM
DDF( 1,J,K)=DDF(    2,J,K)
110     DDF(NI,J,K)=DDF(NIM,J,K)

DO 120 K=2,NKM
DO 120 I=1,NI
DDF(I, 1,K)=DDF(I,    2,K)
120     DDF(I,NJ,K)=DDF(I,NJM,K)

IF(I23.EQ.2) RETURN
DO 130 J=1,NJ
DO 130 I=1,NI
DDF(I,J, 1)=DDF(I,J,    2)
130     DDF(I,J,NK)=DDF(I,J,NKM)

RETURN
END

!***********************************************************************
SUBROUTINE DDF_GET
!***********************************************************************

USE PARAMETERS
USE ICHOS
USE IGRID
USE GRID1
USE GRIDB
USE LEVFN
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL*8 :: FX1,FY1,DX1,DY1,DZ1,DET
!***********************************************************************
IF(IUGRD.EQ.1) RETURN
CALL FUVW_GET
```

```
DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
FX1=(FU(I+1,J,K)-FU(I,J,K))/XC(I)
FY1=(FV(I,J+1,K)-FV(I,J,K))/YC(J)
IF(I23.EQ.3) FZ1=(FW(I,J,K+1)-FW(I,J,K))/ZC(K)
DET=AMAX1(SQRT(FX1**2+FY1**2+FZ1**2),1.E-10)
FX1=FX1/DET
FY1=FY1/DET
FZ1=FZ1/DET
DX1=FX1*XC(I)
DY1=FY1*YC(J)
DZ1=FZ1*ZC(K)
100     DDF(I,J,K)=AMAX1(DTF(I,J,K),SQRT(DX1**2+DY1**2+DZ1**2))

DO 110 K=2,NKM
DO 110 J=2,NJM
DDF( 1,J,K)=DDF(    2,J,K)
110     DDF(NI,J,K)=DDF(NIM,J,K)

DO 120 K=2,NKM
DO 120 I=1,NI
DDF(I, 1,K)=DDF(I,    2,K)
120     DDF(I,NJ,K)=DDF(I,NJM,K)

IF(I23.EQ.2) RETURN
DO 130 J=1,NJ
DO 130 I=1,NI
DDF(I,J, 1)=DDF(I,J,    2)
130     DDF(I,J,NK)=DDF(I,J,NKM)

RETURN
END


!***********************************************************************
SUBROUTINE F_GET
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
```

```
USE UVTPP
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL*8 :: UIW,UIE,VIS,VIN,FLUX1,FLUX2,FLUX3=0,WIB,WIT,FLX,FLY,FLZ
!************************************************************************
CALL FXYZ_GET
!-------------------------------------------------
DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
UIW=U(I   ,J,K)+AMX(I   ,J,K)/RHOF(FU(I   ,J,K))
UIE=U(I+1,J,K)+AMX(I+1,J,K)/RHOF(FU(I+1,J,K))
FLX=0.5*(RU(I)*UIW+RU(I+1)*UIE)/RP(I)

VIS=V(I,J   ,K)+AMY(I,J   ,K)/RHOF(FV(I,J   ,K))
VIN=V(I,J+1,K)+AMY(I,J+1,K)/RHOF(FV(I,J+1,K))
FLY=0.5*(VIS+VIN)-VBUB

FLUX1=AMAX1(FLX,0.)*AW(I,J,K)+AMIN1(FLX,0.)*AE(I,J,K)
FLUX2=AMAX1(FLY,0.)*AS(I,J,K)+AMIN1(FLY,0.)*AN(I,J,K)

IF(I23.EQ.3) THEN
WIB=W(I,J,K   )+AMZ(I,J,K   )/RHOF(FW(I,J,K   ))
WIT=W(I,J,K+1)+AMZ(I,J,K+1)/RHOF(FW(I,J,K+1))
FLZ=0.5*(WIB+WIT)
FLUX3=AMAX1(FLZ,0.)*AB(I,J,K)+AMIN1(FLZ,0.)*AT(I,J,K)
ENDIF

F(I,J,K)=F(I,J,K)-DT*(FLUX1+FLUX2+FLUX3)

100     CONTINUE
!-------------------------------------------------------

CALL F_BC
CALL DDF_GET
CALL IBUB_GET
!
RETURN
END

!************************************************************************
SUBROUTINE F_JMOVE
!************************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
```

```
USE GRIDL
USE LEVFN
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL*8 :: HH,VOLV,VOLH,DVBUB,FLUX,FLY
!***********************************************************************
VOLV=0.
YBUB=0.
DO 10 K=2,NKM
DO 10 J=2,NJM
DO 10 I=2,NIM
HH=0.5+F(I,J,K)/DDF(I,J,K)
HH=AMAX1(0.,AMIN1(1.,HH))
VOLH=(1.-HH)*VOL(I,J,K)*HB(I,J,K)
VOLV=VOLV+VOLH
10      YBUB=YBUB+VOLH*YP(J)
YBUB=YBUB/AMAX1(VOLV,1.E-20)

DVBUB=(YBUB-YBUB0)/DT
VBUB=VBUB+DVBUB
CALL FXYZ_GET

DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
FLY=-DVBUB
FLUX=AMAX1(FLY,0.)*AS(I,J,K)+AMIN1(FLY,0.)*AN(I,J,K)
F(I,J,K)=F(I,J,K)-DT*FLUX
100     CONTINUE
!-------------------------------------------------------
CALL F_BC
CALL DDF_GET
CALL IBUB_GET

RETURN
END

!***********************************************************************
SUBROUTINE F_MOD
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
```

```
USE IDREF
USE SPERR
USE LEVFN
USE RESLT
USE APANS
USE AEANS
USE IBSUB
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,ITER,NFBUB(100),SN,N
REAL*8 :: HH,FX1,FY1,V00,V11,SFMAX,FDS,FSAV,VOLHB
!**********************************************************************
FSAV=F(IREF,JREF,KREF)

ITER=-1
10      ITER=ITER+1
!-------------------------------------------   REINIT
CALL FXYZ_GET
CALL IFFIX_GET

DO 1 K=2,NKM
DO 1 J=2,NJM
DO 1 I=2,NIM
1       CC(I,J,K)=F(I,J,K)/SQRT(F(I,J,K)**2+DTF(I,J,K)**2)

DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
IF(HB(I,J,K).GE.1.E-8 .AND. IFFIX(I,J,K).EQ.1) GOTO 100
SN=SIGN1(CC(I,J,K))
FX1=(AMAX1(0.,SN*AW(I,J,K),-SN*AE(I,J,K)))**2
FY1=(AMAX1(0.,SN*AS(I,J,K),-SN*AN(I,J,K)))**2
IF(I23.EQ.3) FZ1=(AMAX1(0.,SN*AB(I,J,K),-SN*AT(I,J,K)))**2
SS(I,J,K)=CC(I,J,K)*(SQRT(FX1+FY1+FZ1)-1.)
F(I,J,K)=F(I,J,K)-0.5*DTF(I,J,K)*SS(I,J,K)
100     CONTINUE
CALL F_BC

SFMAX=0.
DO 110 K=2,NKM
DO 110 J=2,NJM
DO 110 I=2,NIM
IF(ABS(F(I,J,K)) .GE. 1.5*DDF(I,J,K)) GOTO 110
SFMAX=AMAX1(SFMAX,ABS(SS(I,J,K)))
110     CONTINUE
!---------------------------------------------
IF(IBODY.EQ.1 .AND. ICONB.EQ.1) CALL F_IMMERSED
!********************************* bubbles
CALL IBUB_GET

DO 1200 I2B=N2B,1,-1
```

```
V00=V0H(I2B)
V11=V1H(I2B)
DO 230 N=1,NBUB(I2B)
NFBUB(N)=0
  SURB(N,I2B)=0.
230       VOLB(N,I2B)=0.

DO 240 K=2,NKM
DO 240 J=2,NJM
DO 240 I=2,NIM
N=IBUB(I,J,K,I2B)
IF(N.LT.1) GOTO 240
VOLHB=VOL(I,J,K)*HB(I,J,K)
HH=0.5+F(I,J,K)/DDF(I,J,K)
HH=AMAX1(0.,AMIN1(1.,HH))
VOLB(N,I2B)=VOLB(N,I2B)+(V00-V11*HH)*VOLHB

IF(I2B.EQ.1 .AND. IBUB(I,J,K,2).GE.1) THEN
  IF(IPBUB(IBUB(I,J,K,2),2).EQ.1) GOTO 240
ENDIF
FDS=F(I,J,K)/DDF(I,J,K)
IF(ABS(FDS).LT.1.) SURB(N,I2B)=SURB(N,I2B)&
&                                 +VOLHB*0.5*(1.+COS(PI*FDS))/DDF(I,J,K)
IF(ABS(FDS).LT.1.) NFBUB(N)=NFBUB(N)+1
!        IF(F(I,J,K).LE.0.) NFBUB(N)=NFBUB(N)+1
240     CONTINUE

DO 250 N=1,NBUB(I2B)
IF(NFBUB(N).LE.0) SURB(N,I2B)=0.
IF(SURB(N,I2B).EQ.0.) GOTO 250
SURB(N,I2B)=-V11*(BVOLB(N,I2B)-VOLB(N,I2B))/SURB(N,I2B)
250     CONTINUE

DO 260 K=2,NKM
DO 260 J=2,NJM
DO 260 I=2,NIM
N=IBUB(I,J,K,I2B)
IF(N.LT.1) GOTO 260
IF(I2B.EQ.2) THEN
  IF(IPBUB(IBUB(I,J,K,2),2).EQ.0) GOTO 260
ENDIF
IF(I2B.EQ.1 .AND. IBUB(I,J,K,2).GE.1) THEN
  IF(IPBUB(IBUB(I,J,K,2),2).EQ.1) GOTO 260
ENDIF
F(I,J,K)=F(I,J,K)+SIGN( AMIN1(DDF(I,J,K),ABS(SURB(N,I2B)))&
&                                  ,SURB(N,I2B))
260     CONTINUE
1200    CONTINUE
!
CALL F_BC
CALL IBUB_GET

400     IF(ITER    .EQ. ITER /JMONIT*JMONIT .AND.&
```

```
&     ITIME .EQ. ITIME/JMONT *JMONT)&
&WRITE(I69,4000) ' F',ITER,F(IREF,JREF,KREF)&
&                          ,F(IREF,JREF,KREF)-FSAV,SFMAX,SURB(1,1),SURB(1,2)
IF(ITER .GE. 10)THEN
IF(ITIME .EQ. ITIME/JMONT *JMONT)&
&WRITE(I69,4000) ' F',ITER,F(IREF,JREF,KREF)&
&                          ,F(IREF,JREF,KREF)-FSAV,SFMAX,SURB(1,1),SURB(1,2)
GOTO 20
ENDIF

GOTO 10
20       CONTINUE
CALL DDF_GET
CALL IBUB_GET

4000    FORMAT(A,I6,6(1PE12.4))
END

!***********************************************************************
SUBROUTINE FUVW_GET
!***********************************************************************
USE PARAMETERS
USE IGRID
USE GRIDB
USE LEVFN
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
!***********************************************************************
DO 110 K=K11,KNK
DO 110 J=1,NJ
DO 110 I=2,NI
110     FU(I,J,K)=F(I,J,K)*XDW(I)+F(I-1,J,K)*XDE(I)

DO 120 K=K11,KNK
DO 120 J=2,NJ
DO 120 I=1,NI
120     FV(I,J,K)=F(I,J,K)*YDS(J)+F(I,J-1,K)*YDN(J)

IF(I23.EQ.2) RETURN
DO 130 K=2,NK
DO 130 J=1,NJ
DO 130 I=1,NI
130     FW(I,J,K)=F(I,J,K)*ZDB(K)+F(I,J,K-1)*ZDT(K)

DO 140 K=2,NK
DO 140 J=2,NJ
DO 140 I=2,NI
140     FC(I,J,K)=(FW(I,J    ,K)*XDW(I)+FW(I-1,J    ,K)*XDE(I))*YDS(J)&
&              +(FW(I,J-1,K)*XDW(I)+FW(I-1,J-1,K)*XDE(I))*YDN(J)

RETURN
END
```

```
!***********************************************************************
SUBROUTINE FXYZ_GET
!***********************************************************************
USE PARAMETERS
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE LEVFN
USE APANS
USE AEANS
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
!***********************************************************************
DO 100 K=2,NKM
DO 100 J=2,NJM
DO 100 I=2,NIM
AW(I,J,K)=(F(I,J,K)-F(I-1,J,K))/XD(I)
AE(I,J,K)=(F(I+1,J,K)-F(I,J,K))/XD(I+1)
AC(I,J,K)=(AE(I,J,K)-AW(I,J,K))/(XD(I)+XD(I+1))

AS(I,J,K)=(F(I,J,K)-F(I,J-1,K))/YD(J)
AN(I,J,K)=(F(I,J+1,K)-F(I,J,K))/YD(J+1)
AP(I,J,K)=(AN(I,J,K)-AS(I,J,K))/(YD(J)+YD(J+1))
100     CONTINUE


DO 110 K=2,NKM
DO 110 J=2,NJM
AC( 1,J,K)=0.
110     AC(NI,J,K)=0.

DO 120 K=2,NKM
DO 120 I=2,NIM
AP(I, 1,K)=0.
120     AP(I,NJ,K)=0.

DO 150 K=2,NKM
DO 150 J=2,NJM
DO 150 I=2,NIM
IF(ABS(F(I,J,K)).GT.7.*DDF(I,J,K)) GOTO 150
AW(I,J,K)=AW(I,J,K)+XD(I    )*AMIN2(AC(I,J,K),AC(I-1,J,K))
AE(I,J,K)=AE(I,J,K)-XD(I+1)*AMIN2(AC(I,J,K),AC(I+1,J,K))

AS(I,J,K)=AS(I,J,K)+YD(J    )*AMIN2(AP(I,J,K),AP(I,J-1,K))
AN(I,J,K)=AN(I,J,K)-YD(J+1)*AMIN2(AP(I,J,K),AP(I,J+1,K))
150     CONTINUE
!---------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 300 K=2,NKM
DO 300 J=2,NJM
```

```
DO 300 I=2,NIM
AB(I,J,K)=(F(I,J,K)-F(I,J,K-1))/ZD(K)
AT(I,J,K)=(F(I,J,K+1)-F(I,J,K))/ZD(K+1)
300     AC(I,J,K)=(AT(I,J,K)-AB(I,J,K))/(ZD(K)+ZD(K+1))

DO 310 J=2,NJM
DO 310 I=2,NIM
AC(I,J, 1)=0.
310     AC(I,J,NK)=0.

DO 350 K=2,NKM
DO 350 J=2,NJM
DO 350 I=2,NIM
IF(ABS(F(I,J,K)).GT.7.*DDF(I,J,K)) GOTO 350
AB(I,J,K)=AB(I,J,K)+ZD(K    )*AMIN2(AC(I,J,K),AC(I,J,K-1))
AT(I,J,K)=AT(I,J,K)-ZD(K+1)*AMIN2(AC(I,J,K),AC(I,J,K+1))
350     CONTINUE
RETURN
END
!***********************************************************************
SUBROUTINE H_GET
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE QINTB
USE LEVFN
USE ROGAM
USE RESLT
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,HP,FAC
INTEGER :: ITMP,ITMPE,JTMP,JTMPE,KTMP,KTMPE,NT
REAL*8 :: HH,VOLH,R1
!***********************************************************************
CALL FUVW_GET

DO 100 K=KKST,KKEND !K11,KNK
DO 100 J=JJST,JJEND !1,NJ
DO 100 I=IIST,IIEND !1,NI
HP=1.
IF(F(I,J,K).LE.0.) HP=0.
RHOP(I,J,K)=RO1+(RO2-RO1)*HP
RCP(I,J,K)=RC1+(RC2-RC1)*HP
TKP(I,J,K)=TK1+(TK2-TK1)*HP
```

```
HH=0.5+F(I,J,K)/DDF(I,J,K)
H(I,J,K)=AMAX1(0.,AMIN1(1.,HH))
100     CONTINUE

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(RHOP(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(RHOP(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(RHOP(1,1,1),SIZE(RHOP),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!--------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(RCP(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(RCP(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(
NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(RCP(1,1,1),SIZE(RCP),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!--------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(TKP(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(TKP(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF
```

```
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(TKP(1,1,1),SIZE(TKP),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!---------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(H(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(H(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(H(1,1,1),SIZE(H),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!-------------------------
ITMP=IIST
ITMPE=IIEND
JTMP=JJST
JTMPE=JJEND
KTMP=KKST
KTMPE=KKEND
IF (PIDX.EQ.0) ITMP=IIST+1

DO 110 K=KKST,KKEND !K11,KNK
DO 110 J=JJST,JJEND !1,NJ
DO 110 I=IIST,IIEND !2,NI
RHOU(I,J,K)=RHOP(I,J,K)
IF(T_1PH(F(I,J,K),F(I-1,J,K))) GOTO 110
RHOU(I,J,K)=(F(I,J,K)*RHOP(I,J,K)-F(I-1,J,K)*RHOP(I-1,J,K))/(F(I,J,K)-F(I-1,J,K))
110     CONTINUE

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(RHOU(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(RHOU(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
```

```
CALL
MPI_BCAST(RHOU(1,1,1),SIZE(RHOU),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

ITMP=IIST
ITMPE=IIEND
JTMP=JJST
JTMPE=JJEND
KTMP=KKST
KTMPE=KKEND
IF (PIDY.EQ.0) JTMP=JJST+1

DO 120 K=KKST,KKEND !K11,KNK
DO 120 J=JJST,JJEND !2,NJ
DO 120 I=IIST,IIEND !1,NI
RHOV(I,J,K)=RHOP(I,J,K)
IF(T_1PH(F(I,J,K),F(I,J-1,K))) GOTO 120
RHOV(I,J,K)=(F(I,J,K)*RHOP(I,J,K)-F(I,J-1,K)*RHOP(I,J-1,K))/(F(I,J,K)-F(I,J-1,K))
120     CONTINUE

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(RHOV(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(RHOV(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(RHOV(1,1,1),SIZE(RHOV),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

ITMP=IIST
ITMPE=IIEND
JTMP=JJST
JTMPE=JJEND
KTMP=KKST
KTMPE=KKEND
IF (PIDZ.EQ.0) KTMP=KKST+1

IF(I23.EQ.2) GOTO 135
DO 130 K=KKST,KKEND !2,NK
DO 130 J=JJST,JJEND !1,NJ
DO 130 I=IIST,IIEND !1,NI
RHOW(I,J,K)=RHOP(I,J,K)
IF(T_1PH(F(I,J,K),F(I,J,K-1))) GOTO 130
RHOW(I,J,K)=(F(I,J,K)*RHOP(I,J,K)-F(I,J,K-1)*RHOP(I,J,K-1))/(F(I,J,K)-F(I,J,K-1))
130     CONTINUE
```

```
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(RHOW(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(RHOW(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZ
E(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(RHOW(1,1,1),SIZE(RHOW),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!---------------
135     VOLV=0.
DO 200 K=2,NKM
DO 200 J=2,NJM
DO 200 I=2,NIM
VOLH=(1.-H(I,J,K))*VOL(I,J,K)*HB(I,J,K)
200     VOLV=VOLV+VOLH

IF(I23.EQ.2) THEN
IF(IR.EQ.1) THEN
IF(IBCS.EQ.ISYM) DEFF=(24.*VOLV)**(1./3.)
IF(IBCS.NE.ISYM) DEFF=(12.*VOLV)**(1./3.)
ELSE
IF(IBCS.EQ.ISYM) DEFF=(16.*VOLV/PI)**(1./2.)
IF(IBCS.NE.ISYM) DEFF=( 8.*VOLV/PI)**(1./2.)
ENDIF
ELSE
FAC=1.
IF(IBCW.EQ.ISYM) FAC=FAC*2.
IF(IBCS.EQ.ISYM) FAC=FAC*2.
IF(IBCB.EQ.ISYM) FAC=FAC*2.
DEFF=(6.*FAC*VOLV/PI)**(1./3.)
ENDIF
!-----------------------------------------------------
R0=0.
R1=0.
K=2
DO 250 I=2,NIM
IF( F(I,2,K)*F(I+1,2,K) .LT. 0.) R0=XP(I)-XD(I+1)*F(I,2,K)/(F(I+1,2,K)-F(I,2,K))
IF( F(I,1,K)*F(I+1,1,K) .LT. 0.) R1=XP(I)-XD(I+1)*F(I,1,K)/(F(I+1,1,K)-F(I,1,K))
250     CONTINUE
ANGO=ATAN(YD(2)/AMAX1(EPS,R0-R1))*180./PI

RETURN
END
```

```
!***********************************************************************
SUBROUTINE IN_OUTPUT
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE IBUBS
USE QINTB
USE SPERR
USE LEVFN
USE ULVLL
USE QMSOR
USE RESLT
USE INTXN
USE INTXY
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,II,JJ,KK,N,JFLOUT,JPLOUT,IB1,IB2,ICH,ICH10,ICH100,ICH1000
INTEGER :: JRE=-1,NOUTIN=-1,NOUTI3=-1
REAL*8 :: T0,S0,P0,F0,U1,V1,W1,WI
CHARACTER*5 FILE
!***********************************************************************
ENTRY INT2D_OUT
!***********************************************************************
IF(NBUB(1).GE.1) THEN
IF(I23.EQ.3) CALL INTZY_PLOT(2)
!       IF(I23.EQ.3) CALL INTXZ_PLOT(2)
CALL INTXY_PLOT(2)
ENDIF

CALL QW_GET
CALL QI_GET
IF(IBODY.EQ.1) CALL QBND_GET

NOUTIN=NOUTIN+1
IF(NOUTIN.EQ.0) THEN
OPEN(10,FILE='res1')
OPEN(11,FILE='heat')
OPEN(12,FILE='volb')
OPEN(13,FILE='voll')
OPEN(19,FILE='qwww')
OPEN(21,FILE='i2xy')
OPEN(22,FILE='i2zy')
ELSE
```

```
OPEN(10,FILE='res1',ACCESS='APPEND')
OPEN(11,FILE='heat',ACCESS='APPEND')
OPEN(12,FILE='volb',ACCESS='APPEND')
OPEN(13,FILE='voll',ACCESS='APPEND')
OPEN(19,FILE='qwww',ACCESS='APPEND')
OPEN(21,FILE='i2xy',ACCESS='APPEND')
OPEN(22,FILE='i2zy',ACCESS='APPEND')
ENDIF

IF(IGRAV.EQ.0) THEN
WI=SQRT(ABS(DEFF**2-BDEFF**2)/DT)/2.
WRITE(10,4011) TIME,ITIME,HTOP,R0,    WI,DEFF,VBUB
ELSE
WRITE(10,4011) TIME,ITIME,HTOP,R0,ANGO,DEFF,VBUB
ENDIF
IF(IBODY.EQ.1) THEN
WRITE(11,4011) TIME,ITIME,QWAV,QMAV,QIAV,QIA1,QBND
ELSE
WRITE(11,4011) TIME,ITIME,QWAV,QMAV,QIAV,QIA1,QWS(NIM,NKM)
ENDIF
WRITE(12,4001) TIME,ITIME,NBUB(1),(VOLB(N,1),N=1,NBUB(1))
WRITE(13,4001) TIME,ITIME,NBUB(2),(VOLB(N,2),N=1,NBUB(2))
IF(IBCS.EQ.IWAL) THEN
WRITE(19,4011) TIME,ITIME,QWAV,QMAV,QIAV
WRITE(19,4013) ((QWS(I,K),I=2,NIM),K=2,NKM)
ENDIF
WRITE(21,4012) TIME,ITIME,NPXY,HTOP,R0,ANGO
WRITE(21,4013) ((XZ(I,J),YZ(I,J),J=1,2),I=1,NPXY)
WRITE(22,4012) TIME,ITIME,NPZY,HTOP,R0,ANGO
WRITE(22,4013) ((ZX(I,J),YX(I,J),J=1,2),I=1,NPZY)
RETURN
!***********************************************************************
ENTRY INT3D_OUT
!***********************************************************************
IF(I23.EQ.2) RETURN
IF(NBUB(1) .LT. 1) RETURN

CALL INT3D_PLOT
NOUTI3=NOUTI3+1
IF(NOUTI3 .EQ. 0) THEN
OPEN(20,FILE='i3dd')
ELSE
OPEN(20,FILE='i3dd',ACCESS='APPEND')
ENDIF

WRITE(20,*) NP3D,ITIME
DO 300 I=1,NP3D
WRITE(20,4006) XI(I,1),YI(I,1),ZI(I,1),XI(I,2),YI(I,2),ZI(I,2)
300     WRITE(20,4006) XI(I,3),YI(I,3),ZI(I,3),XI(I,4),YI(I,4),ZI(I,4)
CLOSE(20)
4006    FORMAT(6(1PE12.4))

RETURN
```

```
!***********************************************************************
ENTRY RESTF_IN
!***********************************************************************
OPEN(2,FILE='ccc')
READ(2,*) TIME,ITIME,NBUB(1),NBUB(2),DT,(TIMEW(N),N=1,NCAV)
READ(2,*)
DO 410 K=K11,KNK
DO 410 I=1,NI
DO 410 J=1,NJ
410     READ(2,*) II,JJ,KK,U(I,J,K),V(I,J,K),P(I,J,K),T(I,J,K)
IF(I23.EQ.3) THEN
READ(2,*)
DO 420 K=K11,KNK
DO 420 I=1,NI
DO 420 J=1,NJ
420     READ(2,*) II,JJ,KK,W(I,J,K)
ENDIF
READ(2,*)
DO 430 K=K11,KNK
DO 430 I=1,NI
DO 430 J=1,NJ
430     READ(2,*) II,JJ,KK,IBUB(I,J,K,1),IBUB(I,J,K,2),F(I,J,K)
CLOSE(2)

CALL DDF_GET

RETURN
!***********************************************************************
ENTRY RESTF_OUT
!***********************************************************************
JRE=JRE+1
IF(JRE .EQ. JRE/2*2) OPEN(2,FILE='re1')
IF(JRE .NE. JRE/2*2) OPEN(2,FILE='re2')
WRITE(2,4005) TIME,ITIME,NBUB(1),NBUB(2),DT,(TIMEW(N),N=1,NCAV)
WRITE(2,*) ' I, J, K,U(I,J,K),V(I,J,K),P(I,J,K),T(I,J,K) '
DO 500 K=K11,KNK
DO 500 I=1,NI
DO 500 J=1,NJ
T0=T(I,J,K)
IF(ABS(T0).LE.1.E-99) T0=0.
500     WRITE(2,4002) I,J,K,U(I,J,K),V(I,J,K),P(I,J,K),T0
IF(I23.EQ.3) THEN
WRITE(2,*) ' I, J, K,W(I,J,K)'
DO 510 K=K11,KNK
DO 510 I=1,NI
DO 510 J=1,NJ
510     WRITE(2,4002) I,J,K,W(I,J,K)
ENDIF
WRITE(2,*) ' I, J, K,IBUB(I,J),F(I,J)'
DO 520 K=K11,KNK
DO 520 I=1,NI
DO 520 J=1,NJ
IB1=IBUB(I,J,K,1)
```

```
IB2=IBUB(I,J,K,2)
520     WRITE(2,4003) I,J,K,IB1,IB2,F(I,J,K),S(I,J,K),H(I,J,K),HB(I,J,K)
CLOSE(2)
RETURN



!**********************************************************************
ENTRY OFILE_OUT(JFLOUT)
!**********************************************************************
ICH=ITIME/JFLOUT
ICH1000=ICH/1000
ICH=ICH-ICH1000*1000
ICH100=ICH/100
ICH=ICH-ICH100*100
ICH10=ICH/10
ICH=ICH-ICH10*10
FILE='o'//CHAR(ICH1000+48)//CHAR(ICH100+48)//CHAR(ICH10+48)&
&              //CHAR(ICH+48)
OPEN(2,FILE=FILE)
!
WRITE(2,*) 'TITLE="',TIME,ITIME,DT,""
WRITE(2,4016) TIME,ITIME,DT
WRITE(2,*) 'VARIABLES= "X","Y","Z","U","V","W","T","F","P"'
WRITE(2,*) 'ZONE F=POINT, T="',ITIME,'",I=',NI, ',J=',NJ, ',K=',NK
WRITE(2,4002) NBUB
DO K=K11,KNK
DO J=1,NJ
DO I=1,NI
T0=T(I,J,K); S0=S(I,J,K); P0=P(I,J,K); F0=F(I,J,K)
U1=U(I,J,K); V1=V(I,J,K); W1=W(I,J,K)
IF(I.NE.1 .AND. I.NE.NI) THEN
U1=0.5*(U(I,J,K)+U(I+1,J,K))
END IF
IF(J.NE.1 .AND. J.NE.NJ) THEN
V1=0.5*(V(I,J,K)+V(I,J+1,K))
END IF
IF(K.NE.1 .AND. K.NE.NK) THEN
W1=0.5*(W(I,J,K)+V(I,J,K+1))
END IF
END DO
END DO
END DO
CLOSE(2)
RETURN
!**********************************************************************
ENTRY PLOTF_OUT(JPLOUT)
!**********************************************************************
ICH=ITIME/JPLOUT
ICH1000=ICH/1000
ICH=ICH-ICH1000*1000
ICH100=ICH/100
```

```
ICH=ICH-ICH100*100
ICH10=ICH/10
ICH=ICH-ICH10*10
FILE='p'//CHAR(ICH1000+48)//CHAR(ICH100+48)//CHAR(ICH10+48)&
&             //CHAR(ICH+48)
OPEN(2,FILE=FILE)
K=2
WRITE(2,4005) TIME,ITIME,NBUB(1),NBUB(2),DT,(TIMEW(N),N=1,NCAV)
WRITE(2,*) ' I,J,K,U(I,J,K),V(I,J,K),P(I,J,K),T(I,J,K) '
DO 700 I=1,NI
DO 700 J=1,NJ
T0=T(I,J,K)
IF(ABS(T0).LE.1.E-99) T0=0.
700     WRITE(2,4002) I,J,K,U(I,J,K),V(I,J,K),P(I,J,K),T0
WRITE(2,*) ' I,J,K,IBUB(I,J,K),F(I,J,K) '
DO 710 I=1,NI
DO 710 J=1,NJ
IB1=IBUB(I,J,K,1)
IB2=IBUB(I,J,K,2)
710     WRITE(2,4003) I,J,K,IB1,IB2,F(I,J,K),S(I,J,K),H(I,J,K),HB(I,J,K)
CLOSE(2)
RETURN
!**********************************************************************
4001    FORMAT(1PE13.5,I8,I4,100(1PE12.4))
4005    FORMAT(1PE13.5,I8,I4,I4,100(1PE12.4))
4002    FORMAT(I4,I4,I4,4(1PE17.9))
4003    FORMAT(I4,I4,I4,I4,I4,1(1PE17.9),3(1PE12.4))
4011    FORMAT(1PE13.5,I8,5(1PE12.4))
4012    FORMAT(1PE13.5,I8,I6,3(1PE13.5))
4013    FORMAT(6(1PE13.5))
4016    FORMAT(1PE13.5,I8,1PE13.5)
4018    FORMAT(9(1PE13.5))
!**********************************************************
END

!**********************************************************************
SUBROUTINE INIT
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE INDMO
USE IGRID
USE MUGRD
USE GRID1
USE GRIDB
USE LEVFN
USE ULVLL
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
USE F2F00
USE IBSUB
```

```
USE UVTPP
USE INDIT
USE RELXS
USE X2XOO
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L
REAL*8 :: DIS1,DIS2,DIS3
CHARACTER*3 CHT
COMMON/CHAR1/CHT(6)
!**************************************************************************
NIM=NI-1
NJM=NJ-1
NKM=NK-1
NIMM=NIM-1
NJMM=NJM-1
NKMM=NKM-1

MX(1)=NIM
MY(1)=NJM
MZ(1)=NKM
DO 1 L=2,MULT
IF( (MX(L-1)+1) .NE. (MX(L-1)+1)/2*2 ) THEN
IF (TASKID.EQ.0) WRITE(I69,*) ' CHECK **NI*',L,MX(L-1)+1
STOP
ENDIF
IF( (MY(L-1)+1) .NE. (MY(L-1)+1)/2*2 ) THEN
IF (TASKID.EQ.0) WRITE(I69,*) ' CHECK **NJ*',L,MY(L-1)+1
STOP
ENDIF
IF(I23 .EQ. 3 .AND.    (MZ(L-1)+1) .NE. (MZ(L-1)+1)/2*2 ) THEN
IF (TASKID.EQ.0) WRITE(I69,*) ' CHECK **NK*',L,MZ(L-1)+1
STOP
ENDIF
MX(L)=(MX(L-1)+1)/2
MY(L)=(MY(L-1)+1)/2
MZ(L)=(MZ(L-1)+1)/2
IF(I23 .EQ. 2) MZ(L)=NKM
1       CONTINUE

CALL MULVL
!----------------------------------------INITIALIZE
DO 10 J=1,5
RELAX(J)=1.
10      BRELX(J)=1.

CHT(1)=' U'
CHT(2)=' V'
CHT(3)=' W'
CHT(4)=' T'
CHT(5)=' P'

DO 20 I=1,5
```

```
20       MIT(I)=100
MIT(5)=200

DO 30 K=1,NK
DO 30 J=1,NJ
DO 30 I=1,NI
W(I,J,K)=0.
S(I,J,K)=1.E6
SU(I,J,K)=1.E6
SV(I,J,K)=1.E6
SW(I,J,K)=1.E6
HB(I,J,K)=1.
AB(I,J,K)=0.
AT(I,J,K)=0.
DDF(I,J,K)=DL
DDS(I,J,K)=DL
DTF(I,J,K)=DL
FNZ(I,J,K)=0.
AMZ(I,J,K)=0.
IFFIX(I,J,K)=0
30        MK(I,J,K)=1

DO 40 K=0,NK+1
DO 40 J=0,NJ+1
DO 40 I=0,NI+1
40       FO(I,J,K)=1.E5
!--------------------------------------- GRID
DO 100 I=2,NIM
100     XP(I)=0.5*(XU(I+1)+XU(I))
XP( 1)=XU( 2)
XP(NI)=XU(NI)

DO 110 I=2,NIM
110     XC(I)=XU(I+1)-XU(I)
DO 120 I=2,NI
120     XD(I)=XP(I)-XP(I-1)
DO 122 I=2,NIM
122     XDE(I)=0.5*XC(I)/XD(I)
XDE(NI)=0.
DO 124 I=2,NI
124     XDW(I)=1.-XDE(I)

DO 130 I=2,NI
RU(I)=1.
130     IF(IR .EQ. 1) RU(I)=ABS(XU(I))
DO 140 I=1,NI
RP(I)=1.
140     IF(IR .EQ. 1) RP(I)=ABS(XP(I))

DO 200 J=2,NJM
200     YP(J)=0.5*(YV(J+1)+YV(J))
YP( 1)=YV( 2)
YP(NJ)=YV(NJ)
```

```
DO 210 J=2,NJM
210     YC(J)=YV(J+1)-YV(J)
DO 220 J=2,NJ
220     YD(J)=YP(J)-YP(J-1)
DO 222 J=2,NJM
222     YDN(J)=0.5*YC(J)/YD(J)
YDN(NJ)=0.
DO 224 J=2,NJ
224     YDS(J)=1.-YDN(J)

DO 300 K=2,NKM
300     ZP(K)=0.5*(ZW(K+1)+ZW(K))
ZP( 1)=ZW( 2)
ZP(NK)=ZW(NK)

DO 310 K=2,NKM
310     ZC(K)=ZW(K+1)-ZW(K)
DO 320 K=2,NK
320     ZD(K)=ZP(K)-ZP(K-1)
DO 322 K=2,NKM
322     ZDT(K)=0.5*ZC(K)/ZD(K)
ZDT(NK)=0.
DO 324 K=2,NK
324     ZDB(K)=1.-ZDT(K)

DO 350 K=2,NKM
DO 350 J=2,NJM
DO 350 I=2,NIM
VOL(I,J,K)=RP(I)*XC(I)*YC(J)*ZC(K)
350     CONTINUE

IF(IUGRD.EQ.0) THEN
DO 360 K=2,NKM
DO 360 J=2,NJM
DO 360 I=2,NIM
DIS1=AMAX1(ABS(XD(I)),ABS(XD(I+1)))
DIS2=AMAX1(ABS(YD(J)),ABS(YD(J+1)))
DIS3=AMAX1(ABS(ZD(K)),ABS(ZD(K+1)))
IF(I23.EQ.2) DIS3=1.E10
DTF(I,J,K)=AMIN1(DIS1,DIS2,DIS3)
360     CONTINUE
ENDIF

DO 510 I=1,NI
XPO(I)=XP(I)
510     XDO(I)=XD(I)
XPO(     0)=XPO( 1)
XPO(NI+1)=XPO(NI)
XDO(     1)=0.
XDO(NI+1)=0.

DO 520 J=1,NJ
```

```
YPO(J)=YP(J)
520     YDO(J)=YD(J)
YPO(     0)=YPO( 1)
YPO(NJ+1)=YPO(NJ)
YDO(     1)=0.
YDO(NJ+1)=0.

DO 530 K=1,NK
ZPO(K)=ZP(K)
530     ZDO(K)=ZD(K)
ZPO(     0)=ZPO( 1)
ZPO(NK+1)=ZPO(NK)
ZDO(     1)=0.
ZDO(NK+1)=0.
RETURN
END

!***********************************************************************
SUBROUTINE LNTDMA(N,ITER)
!***********************************************************************
USE PARAMETERS
USE APANS
USE AEANS
USE CALC1
USE MPIVAR
USE IGRID
IMPLICIT NONE
INTEGER ::    NT,II,JJ,KK,I,J,K
INTEGER, INTENT(IN) :: N,ITER
REAL(KIND=8) :: BETA,SSS,GAMA(MD)
INTEGER :: TCOUNT,MPII,MPJJ,MPKK,ONESLICE,TWOSLICE,THREESLICE
!**********************MPI*********************************
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)

CC(:,:,:)=0.

CALL RELOAD

DO K = KKST-1,KKEND+1
DO J = JJST-1,JJEND+1
DO I = IIST-1,IIEND+1
RCC(I,J,K)=0.
ENDDO
ENDDO
ENDDO

!***********************************************************************
DO 999 NT=1,2     !2
!***********************************************************************
IF ((N.EQ.4).OR.(N.EQ.1).OR.(N.EQ.2).OR.(N.EQ.3)) THEN
CALL FRAGMENT
CALL RELOAD
ENDIF
```

```
CALL NEIGHBOUR

IF (N.EQ.4) THEN
CALL SHRINK
CALL RELOAD
ENDIF

IF (N.EQ.1) THEN
CALL CONTRACT(3,2,2,ID-1,JD-1,KD-1)
CALL RELOAD
ENDIF

IF (N.EQ.2) THEN
CALL CONTRACT(2,3,2,ID-1,JD-1,KD-1)
CALL RELOAD
ENDIF

IF (N.EQ.3) THEN
CALL CONTRACT(2,2,3,ID-1,JD-1,KD-1)
CALL RELOAD
ENDIF
!*************************************************************************************************************************
**************
DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND

I=IIST
SSS=SS(I,J,K)+AN(I,J,K)*RCC(I,J+1,K)+AS(I,J,K)*RCC(I,J-1,K)+AT(I,J,K)*RCC(I,J,K+1)+AB(I,J,K)*RCC(
I,J,K-1)
BETA=AP(I,J,K)
RCC(I,J,K)=SSS/BETA

DO 130 I=IIST+1,IIEND
SSS=SS(I,J,K)+AN(I,J,K)*RCC(I,J+1,K)+AS(I,J,K)*RCC(I,J-1,K)+AT(I,J,K)*RCC(I,J,K+1)+AB(I,J,K)*RCC(
I,J,K-1)
GAMA(I)=          -AE(I-1,J,K)/BETA
BETA=AP(I,J,K)+AW(I,J,K)*GAMA(I)
130     RCC(I,J,K)=(SSS+AW(I,J,K)*RCC(I-1,J,K))/BETA

DO 140 I=IIEND-1,IIST,-1
140     RCC(I,J,K)=RCC(I,J,K)-GAMA(I+1)*RCC(I+1,J,K)

100     CONTINUE
!---------------------------------------------------------------
DO 200 K=KKST,KKEND
DO 200 I=IIST,IIEND

J=JJST
SSS=SS(I,J,K)+AE(I,J,K)*RCC(I+1,J,K)+AW(I,J,K)*RCC(I-1,J,K)+AT(I,J,K)*RCC(I,J,K+1)+AB(I,J,K)*RCC
(I,J,K-1)
BETA=AP(I,J,K)
RCC(I,J,K)=SSS/BETA
```

```
DO 230 J=JJST+1,JJEND
SSS=SS(I,J,K)+AE(I,J,K)*RCC(I+1,J,K)+AW(I,J,K)*RCC(I-1,J,K)+AT(I,J,K)*RCC(I,J,K+1)+AB(I,J,K)*RCC
(I,J,K-1)
GAMA(J)=          -AN(I,J-1,K)/BETA
BETA=AP(I,J,K)+AS(I,J,K)*GAMA(J)
230     RCC(I,J,K)=(SSS+AS(I,J,K)*RCC(I,J-1,K))/BETA
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND

K=KKST
SSS=SS(I,J,K)+AE(I,J,K)*RCC(I+1,J,K)+AW(I,J,K)*RCC(I-1,J,K)+AN(I,J,K)*RCC(I,J+1,K)+AS(I,J,K)*RCC
(I,J-1,K)
BETA=AP(I,J,K)
RCC(I,J,K)=SSS/BETA

DO 330 K=KKST,KKEND
SSS=SS(I,J,K)+AE(I,J,K)*RCC(I+1,J,K)+AW(I,J,K)*RCC(I-1,J,K)+AN(I,J,K)*RCC(I,J+1,K)+AS(I,J,K)*RCC
(I,J-1,K)
GAMA(K)=                -AT(I,J,K-1)/BETA
BETA=       AP(I,J,K)+AB(I,J,K)*GAMA(K)
330     RCC(I,J,K)=(SSS+AB(I,J,K)*RCC(I,J,K-1))/BETA

DO 340 K=KKEND-1,KKST,-1
340     RCC(I,J,K)=RCC(I,J,K)-GAMA(K+1)*RCC(I,J,K+1)

300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE

DO K = KKST,KKEND     !Every Processor fills up its own computed part.
DO J = JJST,JJEND
DO I = IIST,IIEND
CC(I,J,K)= RCC(I,J,K)
ENDDO
ENDDO
ENDDO
RETURN
END

!**********************************************************************
SUBROUTINE M1PRTD(I23,IEND,JEND,KEND)
!**********************************************************************
USE PARAMETERS
USE APANS
USE AEANS
USE M1WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
```

```
INTEGER,INTENT(IN)::I23,IEND,JEND,KEND
!***********************************************************************
DO 110 I=IST,IEND
BE(I)=0.
110     BW(I)=0.
DO 112 K=KST,KEND
DO 112 J=JST,JEND
DO 112 I=IST,IEND
BE(I)=BE(I)+AE(I,J,K)
112     BW(I)=BW(I)+AW(I,J,K)
DO 114 I=IST,IEND
114     BX(I)=BE(I)+BW(I)
DO 116 K=KST,KEND
DO 116 I=IST,IEND
116     BX(I)=BX(I)+AS(I,JST,K)+AN(I,JEND,K)
DO 118 J=JST,JEND
DO 118 I=IST,IEND
118     BX(I)=BX(I)+AB(I,J,KST)+AT(I,J,KEND)
!-----------------------------------------------------------------------
DO 120 J=JST,JEND
BN(J)=0.
120     BS(J)=0.
DO 122 K=KST,KEND
DO 122 I=IST,IEND
DO 122 J=JST,JEND
BN(J)=BN(J)+AN(I,J,K)
122     BS(J)=BS(J)+AS(I,J,K)
DO 124 J=JST,JEND
124     BY(J)=BN(J)+BS(J)
DO 126 K=KST,KEND
DO 126 J=JST,JEND
126     BY(J)=BY(J)+AW(IST,J,K)+AE(IEND,J,K)
DO 128 I=IST,IEND
DO 128 J=JST,JEND
128     BY(J)=BY(J)+AB(I,J,KST)+AT(I,J,KEND)
!-----------------------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 130 K=KST,KEND
BT(K)=0.
130     BB(K)=0.
DO 132 J=JST,JEND
DO 132 I=IST,IEND
DO 132 K=KST,KEND
BT(K)=BT(K)+AT(I,J,K)
132     BB(K)=BB(K)+AB(I,J,K)
DO 134 K=KST,KEND
134     BZ(K)=BT(K)+BB(K)
DO 136 J=JST,JEND
DO 136 K=KST,KEND
136     BZ(K)=BZ(K)+AW(IST,J,K)+AE(IEND,J,K)
DO 138 I=IST,IEND
DO 138 K=KST,KEND
138     BZ(K)=BZ(K)+AS(I,JST,K)+AN(I,JEND,K)
```

```
!
RETURN
END
!***********************************************************************
SUBROUTINE M1VTDMA(I23,IEND,JEND,KEND)
!***********************************************************************
USE PARAMETERS
USE CALC1
USE APANS
USE AEANS
USE M1WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT,IST=2,JST=2,KST=2
INTEGER, INTENT(IN)::I23,IEND,JEND,KEND
REAL(KIND=8) :: BETA,SSS
!***********************************************************************
DO 30 I=IST,IEND
30      BLC(I)=0.
DO 32 I=IST,IEND
DO 32 K=KST,KEND
DO 32 J=JST,JEND
32      BLC(I)=BLC(I)+SS(I,J,K)-AP(I,J,K)*CC(I,J,K)&
&                   +AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)&
&                   +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)

BETA=       BX(IST)
BCC(IST)=BLC(IST)/BETA
DO 34 I=IST+1,IEND
GAMA(I)=         -BE(I-1)/BETA
BETA=      BX(I)+BW(I)*GAMA(I)
34      BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=IEND-1,IST,-1
36      BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)

DO 38 I=IST,IEND
DO 38 K=KST,KEND
DO 38 J=JST,JEND
38      CC(I,J,K)=CC(I,J,K)+BCC(I)
!-------------------------------------------------------------
DO 40 J=JST,JEND
40      BLC(J)=0.
DO 42 J=JST,JEND
DO 42 K=KST,KEND
DO 42 I=IST,IEND
42      BLC(J)=BLC(J)+SS(I,J,K)-AP(I,J,K)*CC(I,J,K)&
&                   +AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)&
&                   +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)

BETA=       BY(JST)
```

```
BCC(JST)=BLC(JST)/BETA
DO 43 J=JST+1,JEND
GAMA(J)=          -BN(J-1)/BETA
BETA=       BY(J)+BS(J)*GAMA(J)
43       BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=JEND-1,JST,-1
44       BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)

DO 48 J=JST,JEND
DO 48 K=KST,KEND
DO 48 I=IST,IEND
48       CC(I,J,K)=CC(I,J,K)+BCC(J)
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=KST,KEND
50       BLC(K)=0.
DO 52 K=KST,KEND
DO 52 J=JST,JEND
DO 52 I=IST,IEND
52       BLC(K)=BLC(K)+SS(I,J,K)-AP(I,J,K)*CC(I,J,K)&
&                    +AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                    +AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)&
&                    +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)

BETA=         BZ(KST)
BCC(KST)=BLC(KST)/BETA
DO 54 K=KST+1,KEND
GAMA(K)=-BT(K-1)/BETA
BETA=       BZ(K)+BB(K)*GAMA(K)
54       BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=KEND-1,KST,-1
56       BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)

DO 58 K=KST,KEND
DO 58 J=JST,JEND
DO 58 I=IST,IEND
58       CC(I,J,K)=CC(I,J,K)+BCC(K)

!**********************************************************************
59       DO 999 NT=1,2
!**********************************************************************
DO 100 K=KST,KEND
DO 100 J=JST,JEND
I=IST
SSS=SS(I,J,K)+AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)&
&                    +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)
BETA=AP(I,J,K)
CC(I,J,K)=SSS/BETA

DO 130 I=IST+1,IEND
SSS=SS(I,J,K)+AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)&
&                    +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE(I-1,J,K)/BETA
```

```
BETA=AP(I,J,K)+AW(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW(I,J,K)*CC(I-1,J,K))/BETA

DO 140 I=IEND-1,IST,-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!---------------------------------------------------------------------
DO 200 K=KST,KEND
DO 200 I=IST,IEND
J=JST
SSS=SS(I,J,K)+AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)
BETA=AP(I,J,K)
CC(I,J,K)=SSS/BETA

DO 230 J=JST+1,JEND
SSS=SS(I,J,K)+AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AT(I,J,K)*CC(I,J,K+1)+AB(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN(I,J-1,K)/BETA
BETA=AP(I,J,K)+AS(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS(I,J,K)*CC(I,J-1,K))/BETA

DO 240 J=JEND-1,JST,-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=JST,JEND
DO 300 I=IST,IEND
K=KST
SSS=SS(I,J,K)+AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)
BETA=AP(I,J,K)
CC(I,J,K)=SSS/BETA

DO 330 K=KST+1,KEND
SSS=SS(I,J,K)+AE(I,J,K)*CC(I+1,J,K)+AW(I,J,K)*CC(I-1,J,K)&
&                   +AN(I,J,K)*CC(I,J+1,K)+AS(I,J,K)*CC(I,J-1,K)
GAMA(K)=   -AT(I,J,K-1)/BETA
BETA=      AP(I,J,K)+AB(I,J,K)*GAMA(K)
330     CC(I,J,K)=(SSS+AB(I,J,K)*CC(I,J,K-1))/BETA

DO 340 K=KEND-1,KST,-1
340     CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE

RETURN
END

!**********************************************************************
SUBROUTINE M2PRTD(I23,IEND,JEND,KEND)
```

```
!***********************************************************************
USE PARAMETERS
USE M2APS
USE M2WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
INTEGER,INTENT(IN)::I23,IEND,JEND,KEND
!***********************************************************************
DO 110 I=MXS(2),MXE(2)
BE(I)=0.
110     BW(I)=0.
DO 112 K=MZS(2),MZE(2)
DO 112 J=MYS(2),MYE(2)
DO 112 I=MXS(2),MXE(2)
BE(I)=BE(I)+AE2(I,J,K)
112     BW(I)=BW(I)+AW2(I,J,K)
DO 114 I=MXS(2),MXE(2)
114     BX(I)=BE(I)+BW(I)
DO 116 K=MZS(2),MZE(2)
DO 116 I=MXS(2),MXE(2)
116     BX(I)=BX(I)+AS2(I,MYS(2),K)+AN2(I,MYE(2),K)
DO 118 J=MYS(2),MYE(2)
DO 118 I=MXS(2),MXE(2)
118     BX(I)=BX(I)+AB2(I,J,MZS(2))+AT2(I,J,MZE(2))
!-----------------------------------------------------------------------
DO 120 J=MYS(2),MYE(2)
BN(J)=0.
120     BS(J)=0.
DO 122 K=MZS(2),MZE(2)
DO 122 I=MXS(2),MXE(2)
DO 122 J=MYS(2),MYE(2)
BN(J)=BN(J)+AN2(I,J,K)
122     BS(J)=BS(J)+AS2(I,J,K)
DO 124 J=MYS(2),MYE(2)
124     BY(J)=BN(J)+BS(J)
DO 126 K=MZS(2),MZE(2)
DO 126 J=MYS(2),MYE(2)
126     BY(J)=BY(J)+AW2(MXS(2),J,K)+AE2(MXE(2),J,K)
DO 128 I=MXS(2),MXE(2)
DO 128 J=MYS(2),MYE(2)
128     BY(J)=BY(J)+AB2(I,J,MZS(2))+AT2(I,J,MZE(2))
!-----------------------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 130 K=MZS(2),MZE(2)
BT(K)=0.
130     BB(K)=0.
DO 132 J=MYS(2),MYE(2)
DO 132 I=MXS(2),MXE(2)
DO 132 K=MZS(2),MZE(2)
BT(K)=BT(K)+AT2(I,J,K)
132     BB(K)=BB(K)+AB2(I,J,K)
DO 134 K=MZS(2),MZE(2)
```

```
134     BZ(K)=BT(K)+BB(K)
DO 136 J=MYS(2),MYE(2)
DO 136 K=MZS(2),MZE(2)
136     BZ(K)=BZ(K)+AW2(MXS(2),J,K)+AE2(MXE(2),J,K)
DO 138 I=MXS(2),MXE(2)
DO 138 K=MZS(2),MZE(2)
138     BZ(K)=BZ(K)+AS2(I,MYS(2),K)+AN2(I,MYE(2),K)
!
RETURN
END

!***********************************************************************
SUBROUTINE M2VTDMA(I23,IEND,JEND,KEND)
!***********************************************************************
USE PARAMETERS
USE CALC1, ONLY:CC
USE M2APS
USE M2WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
INTEGER,INTENT(IN)::I23,IEND,JEND,KEND
REAL(KIND=8)::BETA,SSS
!***********************************************************************
DO 30 I=MXS(2),MXE(2)
30      BLC(I)=0.
DO 32 I=MXS(2),MXE(2)
DO 32 K=MZS(2),MZE(2)
DO 32 J=MYS(2),MYE(2)
32      BLC(I)=BLC(I)+SS2(I,J,K)-AP2(I,J,K)*CC(I,J,K)&
&                   +AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                   +AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)&
&                   +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)

BETA=        BX(MXS(2))
BCC(MXS(2))=BLC(MXS(2))/BETA
DO 34 I=MXS(2)+1,MXE(2)
GAMA(I)=          -BE(I-1)/BETA
BETA=       BX(I)+BW(I)*GAMA(I)
34      BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=MXE(2)-1,MXS(2),-1
36      BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)

DO 38 I=MXS(2),MXE(2)
DO 38 K=MZS(2),MZE(2)
DO 38 J=MYS(2),MYE(2)
38      CC(I,J,K)=CC(I,J,K)+BCC(I)
!-------------------------------------------------------------
DO 40 J=MYS(2),MYE(2)
40      BLC(J)=0.
DO 42 J=MYS(2),MYE(2)
DO 42 K=MZS(2),MZE(2)
```

```
DO 42 I=MXS(2),MXE(2)
42      BLC(J)=BLC(J)+SS2(I,J,K)-AP2(I,J,K)*CC(I,J,K)&
&                  +AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                  +AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)&
&                  +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)

BETA=        BY(MYS(2))
BCC(MYS(2))=BLC(MYS(2))/BETA
DO 43 J=MYS(2)+1,MYE(2)
GAMA(J)=          -BN(J-1)/BETA
BETA=       BY(J)+BS(J)*GAMA(J)
43      BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=MYE(2)-1,MYS(2),-1
44      BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)

DO 48 J=MYS(2),MYE(2)
DO 48 K=MZS(2),MZE(2)
DO 48 I=MXS(2),MXE(2)
48      CC(I,J,K)=CC(I,J,K)+BCC(J)
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=MZS(2),MZE(2)
50      BLC(K)=0.
DO 52 K=MZS(2),MZE(2)
DO 52 J=MYS(2),MYE(2)
DO 52 I=MXS(2),MXE(2)
52      BLC(K)=BLC(K)+SS2(I,J,K)-AP2(I,J,K)*CC(I,J,K)&
&                  +AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                  +AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)&
&                  +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)

BETA=        BZ(MZS(2))
BCC(MZS(2))=BLC(MZS(2))/BETA
DO 54 K=MZS(2)+1,MZE(2)
GAMA(K)=   -BT(K-1)/BETA
BETA=       BZ(K)+BB(K)*GAMA(K)
54      BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=MZE(2)-1,MZS(2),-1
56      BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)

DO 58 K=MZS(2),MZE(2)
DO 58 J=MYS(2),MYE(2)
DO 58 I=MXS(2),MXE(2)
58      CC(I,J,K)=CC(I,J,K)+BCC(K)
!**********************************************************************
59      DO 999 NT=1,2
!**********************************************************************
DO 100 K=MZS(2),MZE(2)
DO 100 J=MYS(2),MYE(2)
I=MXS(2)
SSS=SS2(I,J,K)+AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)&
&                  +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)
BETA=AP2(I,J,K)
```

```
CC(I,J,K)=SSS/BETA

DO 130 I=MXS(2)+1,MXE(2)
SSS=SS2(I,J,K)+AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)&
&                   +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE2(I-1,J,K)/BETA
BETA=AP2(I,J,K)+AW2(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW2(I,J,K)*CC(I-1,J,K))/BETA

DO 140 I=MXE(2)-1,MXS(2),-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!---------------------------------------------------------------------
DO 200 K=MZS(2),MZE(2)
DO 200 I=MXS(2),MXE(2)
J=MYS(2)
SSS=SS2(I,J,K)+AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                   +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)
BETA=AP2(I,J,K)
CC(I,J,K)=SSS/BETA

DO 230 J=MYS(2)+1,MYE(2)
SSS=SS2(I,J,K)+AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                   +AT2(I,J,K)*CC(I,J,K+1)+AB2(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN2(I,J-1,K)/BETA
BETA=AP2(I,J,K)+AS2(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS2(I,J,K)*CC(I,J-1,K))/BETA

DO 240 J=MYE(2)-1,MYS(2),-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=MYS(2),MYE(2)
DO 300 I=MXS(2),MXE(2)
K=MZS(2)
SSS=SS2(I,J,K)+AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                   +AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)
BETA=AP2(I,J,K)
CC(I,J,K)=SSS/BETA

DO 330 K=MZS(2)+1,MZE(2)
SSS=SS2(I,J,K)+AE2(I,J,K)*CC(I+1,J,K)+AW2(I,J,K)*CC(I-1,J,K)&
&                   +AN2(I,J,K)*CC(I,J+1,K)+AS2(I,J,K)*CC(I,J-1,K)
GAMA(K)=   -AT2(I,J,K-1)/BETA
BETA=       AP2(I,J,K)+AB2(I,J,K)*GAMA(K)
330     CC(I,J,K)=(SSS+AB2(I,J,K)*CC(I,J,K-1))/BETA

DO 340 K=MZE(2)-1,MZS(2),-1
340     CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE
```

```
IIST=MXS(2)
JJST=MYS(2)
KKST=MZS(2)
IIEND=MXE(2)
JJEND=MYE(2)
KKEND=MZE(2)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE M3PRTD(I23,IEND,JEND,KEND)
!**********************************************************************
USE PARAMETERS
USE M3APS
USE M3WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
!**********************************************************************
DO 110 I=MXS(3),MXE(3)
BE(I)=0.
110     BW(I)=0.
DO 112 K=MZS(3),MZE(3)
DO 112 J=MYS(3),MYE(3)
DO 112 I=MXS(3),MXE(3)
BE(I)=BE(I)+AE3(I,J,K)
112     BW(I)=BW(I)+AW3(I,J,K)
DO 114 I=MXS(3),MXE(3)
114     BX(I)=BE(I)+BW(I)
DO 116 K=MZS(3),MZE(3)
DO 116 I=MXS(3),MXE(3)
116     BX(I)=BX(I)+AS3(I,MYS(3),K)+AN3(I,MYE(3),K)
```

```
DO 118 J=MYS(3),MYE(3)
DO 118 I=MXS(3),MXE(3)
118     BX(I)=BX(I)+AB3(I,J,MZS(3))+AT3(I,J,MZE(3))
!---------------------------------------------------------------------
DO 120 J=MYS(3),MYE(3)
BN(J)=0.
120     BS(J)=0.
DO 122 K=MZS(3),MZE(3)
DO 122 I=MXS(3),MXE(3)
DO 122 J=MYS(3),MYE(3)
BN(J)=BN(J)+AN3(I,J,K)
122     BS(J)=BS(J)+AS3(I,J,K)
DO 124 J=MYS(3),MYE(3)
124     BY(J)=BN(J)+BS(J)
DO 126 K=MZS(3),MZE(3)
DO 126 J=MYS(3),MYE(3)
126     BY(J)=BY(J)+AW3(MXS(3),J,K)+AE3(MXE(3),J,K)
DO 128 I=MXS(3),MXE(3)
DO 128 J=MYS(3),MYE(3)
128     BY(J)=BY(J)+AB3(I,J,MZS(3))+AT3(I,J,MZE(3))
!---------------------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 130 K=MZS(3),MZE(3)
BT(K)=0.
130     BB(K)=0.
DO 132 J=MYS(3),MYE(3)
DO 132 I=MXS(3),MXE(3)
DO 132 K=MZS(3),MZE(3)
BT(K)=BT(K)+AT3(I,J,K)
132     BB(K)=BB(K)+AB3(I,J,K)
DO 134 K=MZS(3),MZE(3)
134     BZ(K)=BT(K)+BB(K)
DO 136 J=MYS(3),MYE(3)
DO 136 K=MZS(3),MZE(3)
136     BZ(K)=BZ(K)+AW3(MXS(3),J,K)+AE3(MXE(3),J,K)
DO 138 I=MXS(3),MXE(3)
DO 138 K=MZS(3),MZE(3)
138     BZ(K)=BZ(K)+AS3(I,MYS(3),K)+AN3(I,MYE(3),K)
!
RETURN
END

!**********************************************************************
SUBROUTINE M3VTDMA(I23,IEND,JEND,KEND)
!**********************************************************************
USE PARAMETERS
USE CALC1,ONLY:CC
USE M3APS
USE M3WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
```

```
INTEGER,INTENT(IN)::I23,IEND,JEND,KEND
REAL(KIND=8)::BETA,SSS
!**********************************************************************
DO 30 I=MXS(3),MXE(3)
30      BLC(I)=0.
DO 32 I=MXS(3),MXE(3)
DO 32 K=MZS(3),MZE(3)
DO 32 J=MYS(3),MYE(3)
32      BLC(I)=BLC(I)+SS3(I,J,K)-AP3(I,J,K)*CC(I,J,K)&
&                   +AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                   +AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)

BETA=       BX(MXS(3))
BCC(MXS(3))=BLC(MXS(3))/BETA
DO 34 I=MXS(3)+1,MXE(3)
GAMA(I)=         -BE(I-1)/BETA
BETA=      BX(I)+BW(I)*GAMA(I)
34      BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=MXE(3)-1,MXS(3),-1
36      BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)

DO 38 I=MXS(3),MXE(3)
DO 38 K=MZS(3),MZE(3)
DO 38 J=MYS(3),MYE(3)
38      CC(I,J,K)=CC(I,J,K)+BCC(I)
!-------------------------------------------------------
DO 40 J=MYS(3),MYE(3)
40      BLC(J)=0.
DO 42 J=MYS(3),MYE(3)
DO 42 K=MZS(3),MZE(3)
DO 42 I=MXS(3),MXE(3)
42      BLC(J)=BLC(J)+SS3(I,J,K)-AP3(I,J,K)*CC(I,J,K)&
&                   +AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                   +AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)

BETA=       BY(MYS(3))
BCC(MYS(3))=BLC(MYS(3))/BETA
DO 43 J=MYS(3)+1,MYE(3)
GAMA(J)=         -BN(J-1)/BETA
BETA=      BY(J)+BS(J)*GAMA(J)
43      BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=MYE(3)-1,MYS(3),-1
44      BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)

DO 48 J=MYS(3),MYE(3)
DO 48 K=MZS(3),MZE(3)
DO 48 I=MXS(3),MXE(3)
48      CC(I,J,K)=CC(I,J,K)+BCC(J)
!-------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=MZS(3),MZE(3)
```

```
50      BLC(K)=0.
DO 52 K=MZS(3),MZE(3)
DO 52 J=MYS(3),MYE(3)
DO 52 I=MXS(3),MXE(3)
52      BLC(K)=BLC(K)+SS3(I,J,K)-AP3(I,J,K)*CC(I,J,K)&
&                   +AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                   +AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)

BETA=         BZ(MZS(3))
BCC(MZS(3))=BLC(MZS(3))/BETA
DO 54 K=MZS(3)+1,MZE(3)
GAMA(K)=   -BT(K-1)/BETA
BETA=       BZ(K)+BB(K)*GAMA(K)
54      BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=MZE(3)-1,MZS(3),-1
56      BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)

DO 58 K=MZS(3),MZE(3)
DO 58 J=MYS(3),MYE(3)
DO 58 I=MXS(3),MXE(3)
58      CC(I,J,K)=CC(I,J,K)+BCC(K)
!************************************************************************
59      DO 999 NT=1,2
!************************************************************************
DO 100 K=MZS(3),MZE(3)
DO 100 J=MYS(3),MYE(3)
I=MXS(3)
SSS=SS3(I,J,K)+AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)
BETA=AP3(I,J,K)
CC(I,J,K)=SSS/BETA

DO 130 I=MXS(3)+1,MXE(3)
SSS=SS3(I,J,K)+AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE3(I-1,J,K)/BETA
BETA=AP3(I,J,K)+AW3(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW3(I,J,K)*CC(I-1,J,K))/BETA

DO 140 I=MXE(3)-1,MXS(3),-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!---------------------------------------------------------------------
DO 200 K=MZS(3),MZE(3)
DO 200 I=MXS(3),MXE(3)
J=MYS(3)
SSS=SS3(I,J,K)+AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                   +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)
BETA=AP3(I,J,K)
CC(I,J,K)=SSS/BETA

DO 230 J=MYS(3)+1,MYE(3)
```

```
SSS=SS3(I,J,K)+AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                    +AT3(I,J,K)*CC(I,J,K+1)+AB3(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN3(I,J-1,K)/BETA
BETA=AP3(I,J,K)+AS3(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS3(I,J,K)*CC(I,J-1,K))/BETA

DO 240 J=MYE(3)-1,MYS(3),-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=MYS(3),MYE(3)
DO 300 I=MXS(3),MXE(3)
K=MZS(3)
SSS=SS3(I,J,K)+AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                    +AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)
BETA=AP3(I,J,K)
CC(I,J,K)=SSS/BETA

DO 330 K=MZS(3)+1,MZE(3)
SSS=SS3(I,J,K)+AE3(I,J,K)*CC(I+1,J,K)+AW3(I,J,K)*CC(I-1,J,K)&
&                    +AN3(I,J,K)*CC(I,J+1,K)+AS3(I,J,K)*CC(I,J-1,K)
GAMA(K)=   -AT3(I,J,K-1)/BETA
BETA=       AP3(I,J,K)+AB3(I,J,K)*GAMA(K)
330     CC(I,J,K)=(SSS+AB3(I,J,K)*CC(I,J,K-1))/BETA

DO 340 K=MZE(3)-1,MZS(3),-1
340     CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE

IIST=MXS(3)
JJST=MYS(3)
KKST=MZS(3)
IIEND=MXE(3)
JJEND=MYE(3)
KKEND=MZE(3)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF
```

```
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!***********************************************************************
SUBROUTINE M4PRTD(I23,IEND,JEND,KEND)
!***********************************************************************
USE PARAMETERS
USE M4APS
USE M4WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
!***********************************************************************
DO 110 I=MXS(4),MXE(4)
BE(I)=0.
110     BW(I)=0.
DO 112 K=MZS(4),MZE(4)
DO 112 J=MYS(4),MYE(4)
DO 112 I=MXS(4),MXE(4)
BE(I)=BE(I)+AE4(I,J,K)
112     BW(I)=BW(I)+AW4(I,J,K)
DO 114 I=MXS(4),MXE(4)
114     BX(I)=BE(I)+BW(I)
DO 116 K=MXS(4),MXE(4)
DO 116 I=MXS(4),MXE(4)
116     BX(I)=BX(I)+AS4(I,MYS(4),K)+AN4(I,MYE(4),K)
DO 118 J=MYS(4),MYE(4)
DO 118 I=MXS(4),MXE(4)
118     BX(I)=BX(I)+AB4(I,J,MZS(4))+AT4(I,J,MZE(4))
!-----------------------------------------------------------------------
DO 120 J=MYS(4),MYE(4)
BN(J)=0.
120     BS(J)=0.
DO 122 K=MZS(4),MZE(4)
DO 122 I=MXS(4),MXE(4)
DO 122 J=MYS(4),MYE(4)
BN(J)=BN(J)+AN4(I,J,K)
122     BS(J)=BS(J)+AS4(I,J,K)
DO 124 J=MYS(4),MYE(4)
124     BY(J)=BN(J)+BS(J)
DO 126 K=MZS(4),MZE(4)
DO 126 J=MYS(4),MYE(4)
126     BY(J)=BY(J)+AW4(MXS(4),J,K)+AE4(MXE(4),J,K)
DO 128 I=MXS(4),MXE(4)
DO 128 J=MYS(4),MYE(4)
128     BY(J)=BY(J)+AB4(I,J,MZS(4))+AT4(I,J,MZE(4))
!-----------------------------------------------------------------------
IF(I23.EQ.2) RETURN
```

```
DO 130 K=MXS(4),MXE(4)
BT(K)=0.
130     BB(K)=0.
DO 132 J=MYS(4),MYE(4)
DO 132 I=MXS(4),MXE(4)
DO 132 K=MZS(4),MZE(4)
BT(K)=BT(K)+AT4(I,J,K)
132     BB(K)=BB(K)+AB4(I,J,K)
DO 134 K=MZS(4),MZE(4)
134     BZ(K)=BT(K)+BB(K)
DO 136 J=MYS(4),MYE(4)
DO 136 K=MZS(4),MZE(4)
136     BZ(K)=BZ(K)+AW4(MXS(4),J,K)+AE4(MXE(4),J,K)
DO 138 I=MXS(4),MXE(4)
DO 138 K=MZS(4),MZE(4) s
138     BZ(K)=BZ(K)+AS4(I,MYS(4),K)+AN4(I,MYE(4),K)
!
RETURN
END

!***********************************************************************
SUBROUTINE M4VTDMA(I23,IEND,JEND,KEND)
!***********************************************************************
USE PARAMETERS
USE CALC1, ONLY:CC
USE M4APS
USE M4WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
REAL(KIND=8):: BETA,SSS
!***********************************************************************
DO 30 I=MXS(4),MXE(4)
30      BLC(I)=0.
DO 32 I=MXS(4),MXE(4)
DO 32 K=MZS(4),MZE(4)
DO 32 J=MYS(4),MYE(4)
32      BLC(I)=BLC(I)+SS4(I,J,K)-AP4(I,J,K)*CC(I,J,K)&
&                   +AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)

BETA=        BX(MXS(4))
BCC(MXS(4))=BLC(MXS(4))/BETA
DO 34 I=MXS(4)+1,MXE(4)
GAMA(I)=          -BE(I-1)/BETA
BETA=       BX(I)+BW(I)*GAMA(I)
34      BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=MXE(4)-1,MXS(4),-1
36      BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)
```

```
DO 38 I=MXS(4),MXE(4)
DO 38 K=MZS(4),MZE(4)
DO 38 J=MYS(4),MYE(4)
38      CC(I,J,K)=CC(I,J,K)+BCC(I)
!-------------------------------------------------------------
DO 40 J=MYS(4),MYE(4)
40      BLC(J)=0.
DO 42 J=MYS(4),MYE(4)
DO 42 K=MZS(4),MZE(4)
DO 42 I=MXS(4),MXE(4)
42      BLC(J)=BLC(J)+SS4(I,J,K)-AP4(I,J,K)*CC(I,J,K)&
&                   +AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
!
BETA=       BY(MYS(4))
BCC(MYS(4))=BLC(MYS(4))/BETA
DO 43 J=MYS(4)+1,MYE(4)
GAMA(J)=         -BN(J-1)/BETA
BETA=      BY(J)+BS(J)*GAMA(J)
43      BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=MYE(4)-1,MYS(4),-1
44      BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)
!
DO 48 J=MYS(4),MYE(4)
DO 48 K=MZS(4),MZE(4)
DO 48 I=MXS(4),MXE(4)
48      CC(I,J,K)=CC(I,J,K)+BCC(J)
!-------------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=MZS(4),MZE(4)
50      BLC(K)=0.
DO 52 K=MZS(4),MZE(4)
DO 52 J=MYS(4),MYE(4)
DO 52 I=MXS(4),MXE(4)
52      BLC(K)=BLC(K)+SS4(I,J,K)-AP4(I,J,K)*CC(I,J,K)&
&                   +AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
!
BETA=       BZ(MZS(4))
BCC(MZS(4))=BLC(MZS(4))/BETA
DO 54 K=MZS(4)+1,MZE(4)
GAMA(K)= -BT(K-1)/BETA
BETA=      BZ(K)+BB(K)*GAMA(K)
54      BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=MZE(4)-1,MZS(4),-1
56      BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)
!
DO 58 K=MZS(4),MZE(4)
DO 58 J=MYS(4),MYE(4)
DO 58 I=MXS(4),MXE(4)
58      CC(I,J,K)=CC(I,J,K)+BCC(K)
```

```
!**********************************************************************
59      DO 999 NT=1,2
!**********************************************************************
DO 100 K=MZS(4),MZE(4)
DO 100 J=MYS(4),MYE(4)
I=MXS(4)
SSS=SS4(I,J,K)+AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
BETA=AP4(I,J,K)
CC(I,J,K)=SSS/BETA
!
DO 130 I=MXS(4)+1,MXE(4)
SSS=SS4(I,J,K)+AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE4(I-1,J,K)/BETA
BETA=AP4(I,J,K)+AW4(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW4(I,J,K)*CC(I-1,J,K))/BETA
!
DO 140 I=MXE(4)-1,MXS(4),-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!----------------------------------------------------------------
DO 200 K=MZS(4),MZE(4)
DO 200 I=MXS(4),MXE(4)
J=MYS(4)
SSS=SS4(I,J,K)+AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
BETA=AP4(I,J,K)
CC(I,J,K)=SSS/BETA
!
DO 230 J=MYS(4)+1,MYE(4)
SSS=SS4(I,J,K)+AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AT4(I,J,K)*CC(I,J,K+1)+AB4(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN4(I,J-1,K)/BETA
BETA=AP4(I,J,K)+AS4(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS4(I,J,K)*CC(I,J-1,K))/BETA
!
DO 240 J=MYE(4)-1,MYS(4),-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!----------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=MYS(4),MYE(4)
DO 300 I=MXS(4),MXE(4)
K=MZS(4)
SSS=SS4(I,J,K)+AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)
BETA=AP4(I,J,K)
CC(I,J,K)=SSS/BETA
!
DO 330 K=MZS(4)+1,MZE(4)
SSS=SS4(I,J,K)+AE4(I,J,K)*CC(I+1,J,K)+AW4(I,J,K)*CC(I-1,J,K)&
&                   +AN4(I,J,K)*CC(I,J+1,K)+AS4(I,J,K)*CC(I,J-1,K)
```

```
GAMA(K)=    -AT4(I,J,K-1)/BETA
BETA=       AP4(I,J,K)+AB4(I,J,K)*GAMA(K)
330     CC(I,J,K)=(SSS+AB4(I,J,K)*CC(I,J,K-1))/BETA
!
DO 340 K=MZE(4)-1,MZS(4),-1
340     CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE

IIST=MXS(4)
JJST=MYS(4)
KKST=MZS(4)
IIEND=MXE(4)
JJEND=MYE(4)
KKEND=MZE(4)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE M5PRTD(I23,IEND,JEND,KEND)
!**********************************************************************
USE PARAMETERS
USE M5APS
USE M5WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
!**********************************************************************
DO 110 I=MXS(5),MXE(5)
BE(I)=0.
110     BW(I)=0.
DO 112 K=MZS(5),MZE(5)
```

```
DO 112 J=MYS(5),MYE(5)
DO 112 I=MXS(5),MXE(5)
BE(I)=BE(I)+AE5(I,J,K)
112     BW(I)=BW(I)+AW5(I,J,K)
DO 114 I=MXS(5),MXE(5)
114     BX(I)=BE(I)+BW(I)
DO 116 K=MZS(5),MZE(5)
DO 116 I=MXS(5),MXE(5)
116     BX(I)=BX(I)+AS5(I,MYS(5),K)+AN5(I,MYE(5),K)
DO 118 J=MYS(5),MYE(5)
DO 118 I=MXS(5),MXE(5)
118     BX(I)=BX(I)+AB5(I,J,MZS(5))+AT5(I,J,MZE(5))
!---------------------------------------------------------------------
DO 120 J=MYS(5),MYE(5)
BN(J)=0.
120     BS(J)=0.
DO 122 K=MZS(5),MZE(5)
DO 122 I=MXS(5),MXE(5)
DO 122 J=MYS(5),MYE(5)
BN(J)=BN(J)+AN5(I,J,K)
122     BS(J)=BS(J)+AS5(I,J,K)
DO 124 J=MYS(5),MYE(5)
124     BY(J)=BN(J)+BS(J)
DO 126 K=MZS(5),MZE(5)
DO 126 J=MYS(5),MYE(5)
126     BY(J)=BY(J)+AW5(MXS(5),J,K)+AE5(MXE(5),J,K)
DO 128 I=MXS(5),MXE(5)
DO 128 J=MYS(5),MYE(5)
128     BY(J)=BY(J)+AB5(I,J,MZS(5))+AT5(I,J,MZE(5))
!---------------------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 130 K=MZS(5),MZE(5)
BT(K)=0.
130     BB(K)=0.
DO 132 J=MYS(5),MYE(5)
DO 132 I=MXS(5),MXE(5)
DO 132 K=MZS(5),MZE(5)
BT(K)=BT(K)+AT5(I,J,K)
132     BB(K)=BB(K)+AB5(I,J,K)
DO 134 K=MZS(5),MZE(5)
134     BZ(K)=BT(K)+BB(K)
DO 136 J=MYS(5),MYE(5)
DO 136 K=MZS(5),MZE(5)
136     BZ(K)=BZ(K)+AW5(MXS(5),J,K)+AE5(MXE(5),J,K)
DO 138 I=MXS(5),MXE(5)
DO 138 K=MZS(5),MZE(5)
138     BZ(K)=BZ(K)+AS5(I,MYS(5),K)+AN5(I,MYE(5),K)
!
RETURN
END

!**********************************************************************
SUBROUTINE M5VTDMA(I23,IEND,JEND,KEND)
```

```
!***********************************************************************
USE PARAMETERS
USE CALC1,ONLY:CC
USE M5APS
USE M5WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
REAL(KIND=8):: BETA,SSS
!***********************************************************************
DO 30 I=MXS(5),MXE(5)
30      BLC(I)=0.
DO 32 I=MXS(5),MXE(5)
DO 32 K=MZS(5),MZE(5)
DO 32 J=MYS(5),MYE(5)
32      BLC(I)=BLC(I)+SS5(I,J,K)-AP5(I,J,K)*CC(I,J,K)&
&                   +AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                   +AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)&
&                   +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)

BETA=         BX(MXS(5))
BCC(MXS(5))=BLC(MXS(5))/BETA
DO 34 I=MXS(5)+1,MXE(5)
GAMA(I)=          -BE(I-1)/BETA
BETA=       BX(I)+BW(I)*GAMA(I)
34      BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=MXE(5)-1,MXS(5),-1
36      BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)

DO 38 I=MXS(5),MXE(5)
DO 38 K=MZS(5),MZE(5)
DO 38 J=MYS(5),MYE(5)
38      CC(I,J,K)=CC(I,J,K)+BCC(I)
!-------------------------------------------------------------
DO 40 J=MYS(5),MYE(5)
40      BLC(J)=0.
DO 42 J=MYS(5),MYE(5)
DO 42 K=MZS(5),MZE(5)
DO 42 I=MXS(5),MXE(5)
42      BLC(J)=BLC(J)+SS5(I,J,K)-AP5(I,J,K)*CC(I,J,K)&
&                   +AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                   +AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)&
&                   +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)
!
BETA=         BY(MYS(5))
BCC(MYS(5))=BLC(MYS(5))/BETA
DO 43 J=MYS(5)+1,MYE(5)
GAMA(J)=          -BN(J-1)/BETA
BETA=       BY(J)+BS(J)*GAMA(J)
43      BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=MYE(5)-1,MYS(5),-1
```

```
44      BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)

DO 48 J=MYS(5),MYE(5)
DO 48 K=MZS(5),MZE(5)
DO 48 I=MXS(5),MXE(5)
48      CC(I,J,K)=CC(I,J,K)+BCC(J)
!---------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=MZS(5),MZE(5)
50      BLC(K)=0.
DO 52 K=MZS(5),MZE(5)
DO 52 J=MYS(5),MYE(5)
DO 52 I=MXS(5),MXE(5)
52      BLC(K)=BLC(K)+SS5(I,J,K)-AP5(I,J,K)*CC(I,J,K)&
&                   +AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                   +AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)&
&                   +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)

BETA=        BZ(MZS(5))
BCC(MZS(5))=BLC(MZS(5))/BETA
DO 54 K=MZS(5)+1,MZE(5)
GAMA(K)=-BT(K-1)/BETA
BETA=      BZ(K)+BB(K)*GAMA(K)
54      BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=MZE(5)-1,MZS(5),-1
56      BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)

DO 58 K=MZS(5),MZE(5)
DO 58 J=MYS(5),MYE(5)
DO 58 I=MXS(5),MXE(5)
58      CC(I,J,K)=CC(I,J,K)+BCC(K)
!*********************************************************************
59      DO 999 NT=1,2
!*********************************************************************
DO 100 K=MZS(5),MZE(5)
DO 100 J=MYS(5),MYE(5)
I=MXS(5)
SSS=SS5(I,J,K)+AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)&
&                   +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)
BETA=AP5(I,J,K)
CC(I,J,K)=SSS/BETA

DO 130 I=MXS(5)+1,MXE(5)
SSS=SS5(I,J,K)+AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)&
&                   +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE5(I-1,J,K)/BETA
BETA=AP5(I,J,K)+AW5(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW5(I,J,K)*CC(I-1,J,K))/BETA

DO 140 I=MXE(5)-1,MXS(5),-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!---------------------------------------------------------------
```

```
DO 200 K=MZS(5),MZE(5)
DO 200 I=MXS(5),MXE(5)
J=MYS(5)
SSS=SS5(I,J,K)+AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                    +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)
BETA=AP5(I,J,K)
CC(I,J,K)=SSS/BETA
!
DO 230 J=MYS(5)+1,MYE(5)
SSS=SS5(I,J,K)+AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                    +AT5(I,J,K)*CC(I,J,K+1)+AB5(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN5(I,J-1,K)/BETA
BETA=AP5(I,J,K)+AS5(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS5(I,J,K)*CC(I,J-1,K))/BETA

DO 240 J=MYE(5)-1,MYS(5),-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=MYS(5),MYE(5)
DO 300 I=MXS(5),MXE(5)
K=MZS(5)
SSS=SS5(I,J,K)+AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                    +AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)
BETA=AP5(I,J,K)
CC(I,J,K)=SSS/BETA

DO 330 K=MZS(5)+1,MZE(5)
SSS=SS5(I,J,K)+AE5(I,J,K)*CC(I+1,J,K)+AW5(I,J,K)*CC(I-1,J,K)&
&                    +AN5(I,J,K)*CC(I,J+1,K)+AS5(I,J,K)*CC(I,J-1,K)
GAMA(K)=    -AT5(I,J,K-1)/BETA
BETA=       AP5(I,J,K)+AB5(I,J,K)*GAMA(K)
330     CC(I,J,K)=(SSS+AB5(I,J,K)*CC(I,J,K-1))/BETA

DO 340 K=MZE(5)-1,MZS(5),-1
340     CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300     CONTINUE
!---------------------------------------------------------------------
999     CONTINUE

IIST=MXS(5)
JJST=MYS(5)
KKST=MZS(5)
IIEND=MXE(5)
JJEND=MYE(5)
KKEND=MZE(5)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
```

```
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE M6PRTD(I23,IEND,JEND,KEND)
!**********************************************************************
USE PARAMETERS
USE M6APS
USE M6WRK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
!**********************************************************************
DO 110 I=MXS(6),MXE(6)
BE(I)=0.
110     BW(I)=0.
DO 112 K=MZS(6),MZE(6)
DO 112 J=MYS(6),MYE(6)
DO 112 I=MXS(6),MXE(6)
BE(I)=BE(I)+AE6(I,J,K)
112     BW(I)=BW(I)+AW6(I,J,K)
DO 114 I=MXS(6),MXE(6)
114     BX(I)=BE(I)+BW(I)
DO 116 K=MZS(6),MZE(6)
DO 116 I=MXS(6),MXE(6)
116     BX(I)=BX(I)+AS6(I,MYS(6),K)+AN6(I,MYE(6),K)
DO 118 J=MYS(6),MYE(6)
DO 118 I=MXS(6),MXE(6)
118     BX(I)=BX(I)+AB6(I,J,MZS(6))+AT6(I,J,MZE(6))
!----------------------------------------------------------------------
DO 120 J=MYS(6),MYE(6)
BN(J)=0.
120     BS(J)=0.
DO 122 K=MZS(6),MZE(6)
DO 122 I=MXS(6),MXE(6)
DO 122 J=MYS(6),MYE(6)
BN(J)=BN(J)+AN6(I,J,K)
122     BS(J)=BS(J)+AS6(I,J,K)
DO 124 J=MYS(6),MYE(6)
```

```
124     BY(J)=BN(J)+BS(J)
DO 126 K=MZS(6),MZE(6)
DO 126 J=MYS(6),MYE(6)
126     BY(J)=BY(J)+AW6(MXS(6),J,K)+AE6(MXE(6),J,K)
DO 128 I=MXS(6),MXE(6)
DO 128 J=MYS(6),MYE(6)
128     BY(J)=BY(J)+AB6(I,J,MZS(6))+AT6(I,J,MZE(6))
!-----------------------------------------------------------------------
IF(I23.EQ.2) RETURN
DO 130 K=MZS(6),MZE(6)
BT(K)=0.
130     BB(K)=0.
DO 132 J=MYS(6),MYE(6)
DO 132 I=MXS(6),MXE(6)
DO 132 K=MZS(6),MZE(6)
BT(K)=BT(K)+AT6(I,J,K)
132     BB(K)=BB(K)+AB6(I,J,K)
DO 134 K=MZS(6),MZE(6)
134     BZ(K)=BT(K)+BB(K)
DO 136 J=MYS(6),MYE(6)
DO 136 K=MZS(6),MZE(6)
136     BZ(K)=BZ(K)+AW6(MXS(6),J,K)+AE6(MXE(6),J,K)
DO 138 I=MXS(6),MXE(6)
DO 138 K=MZS(6),MZE(6)
138     BZ(K)=BZ(K)+AS6(I,MYS(6),K)+AN6(I,MYE(6),K)
!
RETURN
END

!***********************************************************************
SUBROUTINE M6VTDMA(I23,IEND,JEND,KEND)
!***********************************************************************
USE PARAMETERS
USE CALC1,ONLY:CC
USE M6APS
USE M6WRK
USE TDWK2
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
INTEGER,INTENT(IN):: I23,IEND,JEND,KEND
REAL(KIND=8):: BETA,SSS
!***********************************************************************
DO 30 I=MXS(6),MXE(6)
30      BLC(I)=0.
DO 32 I=MXS(6),MXE(6)
DO 32 K=MZS(6),MZE(6)
DO 32 J=MYS(6),MYE(6)
32      BLC(I)=BLC(I)+SS6(I,J,K)-AP6(I,J,K)*CC(I,J,K)&
&                   +AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                   +AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)&
&                   +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)
!
```

```
BETA=        BX(MXS(6))
BCC(MXS(6))=BLC(MXS(6))/BETA
DO 34 I=MXS(6)+1,MXE(6)
GAMA(I)=          -BE(I-1)/BETA
BETA=       BX(I)+BW(I)*GAMA(I)
34       BCC(I)=(BLC(I)+BW(I)*BCC(I-1))/BETA
DO 36 I=MXE(6)-1,MXS(6),-1
36       BCC(I)=BCC(I)-GAMA(I+1)*BCC(I+1)
!
DO 38 I=MXS(6),MXE(6)
DO 38 K=MZS(6),MZE(6)
DO 38 J=MYS(6),MYE(6)
38       CC(I,J,K)=CC(I,J,K)+BCC(I)
!------------------------------------------------------------
DO 40 J=MYS(6),MYE(6)
40       BLC(J)=0.
DO 42 J=MYS(6),MYE(6)
DO 42 K=MZS(6),MZE(6)
DO 42 I=MXS(6),MXE(6)
42       BLC(J)=BLC(J)+SS6(I,J,K)-AP6(I,J,K)*CC(I,J,K)&
&                    +AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                    +AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)&
&                    +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)

BETA=        BY(MYS(6))
BCC(MYS(6))=BLC(MYS(6))/BETA
DO 43 J=MYS(6)+1,MYE(6)
GAMA(J)=          -BN(J-1)/BETA
BETA=       BY(J)+BS(J)*GAMA(J)
43       BCC(J)=(BLC(J)+BS(J)*BCC(J-1))/BETA
DO 44 J=MYE(6)-1,MYS(6),-1
44       BCC(J)=BCC(J)-GAMA(J+1)*BCC(J+1)

DO 48 J=MYS(6),MYE(6)
DO 48 K=MZS(6),MZE(6)
DO 48 I=MXS(6),MXE(6)
48       CC(I,J,K)=CC(I,J,K)+BCC(J)
!------------------------------------------------------------
IF(I23.EQ.2) GOTO 59
DO 50 K=MZS(6),MZE(6)
50       BLC(K)=0.
DO 52 K=MZS(6),MZE(6)
DO 52 J=MYS(6),MYE(6)
DO 52 I=MXS(6),MXE(6)
52       BLC(K)=BLC(K)+SS6(I,J,K)-AP6(I,J,K)*CC(I,J,K)&
&                    +AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                    +AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)&
&                    +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)

BETA=        BZ(MZS(6))
BCC(MZS(6))=BLC(MZS(6))/BETA
DO 54 K=MZS(6)+1,MZE(6)
GAMA(K)=    -BT(K-1)/BETA
```

```
BETA=       BZ(K)+BB(K)*GAMA(K)
54      BCC(K)=(BLC(K)+BB(K)*BCC(K-1))/BETA
DO 56 K=MZE(6)-1,MZS(6),-1
56      BCC(K)=BCC(K)-GAMA(K+1)*BCC(K+1)

DO 58 K=MZS(6),MZE(6)
DO 58 J=MYS(6),MYE(6)
DO 58 I=MXS(6),MXE(6)
58      CC(I,J,K)=CC(I,J,K)+BCC(K)
!************************************************************************
59      DO 999 NT=1,2
!************************************************************************
DO 100 K=MZS(6),MZE(6)
DO 100 J=MYS(6),MYE(6)
I=MXS(6)
SSS=SS6(I,J,K)+AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)&
&                   +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)
BETA=AP6(I,J,K)
CC(I,J,K)=SSS/BETA

DO 130 I=MXS(6)+1,MXE(6)
SSS=SS6(I,J,K)+AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)&
&                   +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)
GAMA(I)=          -AE6(I-1,J,K)/BETA
BETA=AP6(I,J,K)+AW6(I,J,K)*GAMA(I)
130     CC(I,J,K)=(SSS+AW6(I,J,K)*CC(I-1,J,K))/BETA

DO 140 I=MXE(6)-1,MXS(6),-1
140     CC(I,J,K)=CC(I,J,K)-GAMA(I+1)*CC(I+1,J,K)
100     CONTINUE
!---------------------------------------------------------------------
DO 200 K=MZS(6),MZE(6)
DO 200 I=MXS(6),MXE(6)
J=MYS(6)
SSS=SS6(I,J,K)+AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                   +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)
BETA=AP6(I,J,K)
CC(I,J,K)=SSS/BETA

DO 230 J=MYS(6)+1,MYE(6)
SSS=SS6(I,J,K)+AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                   +AT6(I,J,K)*CC(I,J,K+1)+AB6(I,J,K)*CC(I,J,K-1)
GAMA(J)=          -AN6(I,J-1,K)/BETA
BETA=AP6(I,J,K)+AS6(I,J,K)*GAMA(J)
230     CC(I,J,K)=(SSS+AS6(I,J,K)*CC(I,J-1,K))/BETA

DO 240 J=MYE(6)-1,MYS(6),-1
240     CC(I,J,K)=CC(I,J,K)-GAMA(J+1)*CC(I,J+1,K)
200     CONTINUE
!---------------------------------------------------------------------
IF(I23.EQ.2) GOTO 999
DO 300 J=MYS(6),MYE(6)
DO 300 I=MXS(6),MXE(6)
```

```
K=MZS(6)
SSS=SS6(I,J,K)+AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                   +AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)
BETA=AP6(I,J,K)
CC(I,J,K)=SSS/BETA

DO 330 K=MZS(6)+1,MZE(6)
SSS=SS6(I,J,K)+AE6(I,J,K)*CC(I+1,J,K)+AW6(I,J,K)*CC(I-1,J,K)&
&                   +AN6(I,J,K)*CC(I,J+1,K)+AS6(I,J,K)*CC(I,J-1,K)
GAMA(K)=   -AT6(I,J,K-1)/BETA
BETA=      AP6(I,J,K)+AB6(I,J,K)*GAMA(K)
330    CC(I,J,K)=(SSS+AB6(I,J,K)*CC(I,J,K-1))/BETA

DO 340 K=MZE(6)-1,MZS(6),-1
340    CC(I,J,K)=CC(I,J,K)-GAMA(K+1)*CC(I,J,K+1)
300    CONTINUE
!---------------------------------------------------------------------
999    CONTINUE

IIST=MXS(6)
JJST=MYS(6)
KKST=MZS(6)
IIEND=MXE(6)
JJEND=MYE(6)
KKEND=MZE(6)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN

END

!**********************************************************************
SUBROUTINE MU_GEAP
!**********************************************************************
USE PARAMETERS
USE IGRID
```

```
USE MUGRD
USE APANS
USE AEANS
USE M2APS
USE M3APS
USE M4APS
USE M5APS
USE M6APS
USE MPIVAR
IMPLICIT NONE
INTEGER :: L,I,J,K,MZNI,NT
!**************************************************************************
MZNI=2
!
L=2
IF(I23 .EQ. 3) MZNI=MZ(L)
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AW2(I,J,K)=0.5*( AW(2*I-2,2*J-1,2*K-2)+AW(2*I-2,2*J-2,2*K-2)+AW(2*I-2,2*J-1,2*K-1)+AW(2*I-2,2*J-2,2*
K-1))
AE2(I,J,K)=0.5*( AE(2*I-1,2*J-1,2*K-2)+AE(2*I-1,2*J-2,2*K-2)+AE(2*I-1,2*J-1,2*K-1)+AE(2*I-1,2*J-2,2*K-
1))
AS2(I,J,K)=0.5*( AS(2*I-1,2*J-2,2*K-2)+AS(2*I-2,2*J-2,2*K-2)+AS(2*I-1,2*J-2,2*K-1)+AS(2*I-2,2*J-2,2*K-
1))
AN2(I,J,K)=0.5*( AN(2*I-1,2*J-1,2*K-2)+AN(2*I-2,2*J-1,2*K-2)+AN(2*I-1,2*J-1,2*K-1)+AN(2*I-2,2*J-1,2*K-
1))
AT2(I,J,K)=0.5*( AT(2*I-1,2*J-2,2*K-1)+AT(2*I-2,2*J-2,2*K-1)+AT(2*I-1,2*J-1,2*K-1)+AT(2*I-2,2*J-1,2*K-1
))
AB2(I,J,K)=0.5*( AB(2*I-1,2*J-2,2*K-2)+AB(2*I-2,2*J-2,2*K-2)+AB(2*I-1,2*J-1,2*K-2)+AB(2*I-2,2*J-1,2*K-
2))
ENDDO
ENDDO
ENDDO
!
DO K= MZS(L),MZE(L) !2,MZ(L)
DO J= MYS(L),MYE(L) !2,MY(L)
DO I= MXS(L),MXE(L) !2,MX(L)
AP2(I,J,K)=AE2(I,J,K)+AN2(I,J,K)+AT2(I,J,K)+AW2(I,J,K)+AS2(I,J,K)+AB2(I,J,K)
ENDDO
ENDDO
ENDDO
!
IF(MULT .EQ. 2) RETURN
!--------------------------------------------------------------------------
L=3
IF(I23 .EQ. 3) MZNI=MZ(L)
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AW3(I,J,K)=0.5*( AW2(2*I-2,2*J-1,2*K-2)+AW2(2*I-2,2*J-2,2*K-2)+AW2(2*I-2,2*J-1,2*K-1)+AW2(2*I-2,2*
J-2,2*K-1))
```

```
AE3(I,J,K)=0.5*( AE2(2*I-1,2*J-1,2*K-2)+AE2(2*I-1,2*J-2,2*K-2)+AE2(2*I-1,2*J-1,2*K-1)+AE2(2*I-1,2*J-2,
2*K-1))
AS3(I,J,K)=0.5*( AS2(2*I-1,2*J-2,2*K-2)+AS2(2*I-2,2*J-2,2*K-2)+AS2(2*I-1,2*J-2,2*K-1)+AS2(2*I-2,2*J-2,
2*K-1))
AN3(I,J,K)=0.5*( AN2(2*I-1,2*J-1,2*K-2)+AN2(2*I-2,2*J-1,2*K-2)+AN2(2*I-1,2*J-1,2*K-1)+AN2(2*I-2,2*J-1
,2*K-1))
AT3(I,J,K)=0.5*( AT2(2*I-1,2*J-2,2*K-1)+AT2(2*I-2,2*J-2,2*K-1)+AT2(2*I-1,2*J-1,2*K-1)+AT2(2*I-2,2*J-1,
2*K-1))
AB3(I,J,K)=0.5*( AB2(2*I-1,2*J-2,2*K-2)+AB2(2*I-2,2*J-2,2*K-2)+AB2(2*I-1,2*J-1,2*K-2)+AB2(2*I-2,2*J-1,
2*K-2))
ENDDO
ENDDO
ENDDO
!
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AP3(I,J,K)=AE3(I,J,K)+AN3(I,J,K)+AT3(I,J,K)+AW3(I,J,K)+AS3(I,J,K)+AB3(I,J,K)
ENDDO
ENDDO
ENDDO
!
IF(MULT .EQ. 3) RETURN
!---------------------------------------------------------------------
L=4
IF(I23 .EQ. 3) MZNI=MZ(L)
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AW4(I,J,K)=0.5*( AW3(2*I-2,2*J-1,2*K-2)+AW3(2*I-2,2*J-2,2*K-2)+AW3(2*I-2,2*J-1,2*K-1)+AW3(2*I-2,2*
J-2,2*K-1))
AE4(I,J,K)=0.5*( AE3(2*I-1,2*J-1,2*K-2)+AE3(2*I-1,2*J-2,2*K-2)+AE3(2*I-1,2*J-1,2*K-1)+AE3(2*I-1,2*J-2,
2*K-1))
AS4(I,J,K)=0.5*( AS3(2*I-1,2*J-2,2*K-2)+AS3(2*I-2,2*J-2,2*K-2)+AS3(2*I-1,2*J-2,2*K-1)+AS3(2*I-2,2*J-2,
2*K-1))
AN4(I,J,K)=0.5*( AN3(2*I-1,2*J-1,2*K-2)+AN3(2*I-2,2*J-1,2*K-2)+AN3(2*I-1,2*J-1,2*K-1)+AN3(2*I-2,2*J-1
,2*K-1))
AT4(I,J,K)=0.5*( AT3(2*I-1,2*J-2,2*K-1)+AT3(2*I-2,2*J-2,2*K-1)+AT3(2*I-1,2*J-1,2*K-1)+AT3(2*I-2,2*J-1,
2*K-1))
AB4(I,J,K)=0.5*( AB3(2*I-1,2*J-2,2*K-2)+AB3(2*I-2,2*J-2,2*K-2)+AB3(2*I-1,2*J-1,2*K-2)+AB3(2*I-2,2*J-1,
2*K-2))
ENDDO
ENDDO
ENDDO
!
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AP4(I,J,K)=AE4(I,J,K)+AN4(I,J,K)+AT4(I,J,K)+AW4(I,J,K)+AS4(I,J,K)+AB4(I,J,K)
ENDDO
ENDDO
ENDDO
!
```

```
IF(MULT .EQ. 4) RETURN
!---------------------------------------------------------------------
L=5
IF(I23 .EQ. 3) MZNI=MZ(L)
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AW5(I,J,K)=0.5*( AW4(2*I-2,2*J-1,2*K-2)+AW4(2*I-2,2*J-2,2*K-2)+AW4(2*I-2,2*J-1,2*K-1)+AW4(2*I-2,2*
J-2,2*K-1))
AE5(I,J,K)=0.5*( AE4(2*I-1,2*J-1,2*K-2)+AE4(2*I-1,2*J-2,2*K-2)+AE4(2*I-1,2*J-1,2*K-1)+AE4(2*I-1,2*J-2,
2*K-1))
AS5(I,J,K)=0.5*( AS4(2*I-1,2*J-2,2*K-2)+AS4(2*I-2,2*J-2,2*K-2)+AS4(2*I-1,2*J-2,2*K-1)+AS4(2*I-2,2*J-2,
2*K-1))
AN5(I,J,K)=0.5*( AN4(2*I-1,2*J-1,2*K-2)+AN4(2*I-2,2*J-1,2*K-2)+AN4(2*I-1,2*J-1,2*K-1)+AN4(2*I-2,2*J-1
,2*K-1))
AT5(I,J,K)=0.5*( AT4(2*I-1,2*J-2,2*K-1)+AT4(2*I-2,2*J-2,2*K-1)+AT4(2*I-1,2*J-1,2*K-1)+AT4(2*I-2,2*J-1,
2*K-1))
AB5(I,J,K)=0.5*( AB4(2*I-1,2*J-2,2*K-2)+AB4(2*I-2,2*J-2,2*K-2)+AB4(2*I-1,2*J-1,2*K-2)+AB4(2*I-2,2*J-1,
2*K-2))
ENDDO
ENDDO
ENDDO
!
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AP5(I,J,K)=AE5(I,J,K)+AN5(I,J,K)+AT5(I,J,K)+AW5(I,J,K)+AS5(I,J,K)+AB5(I,J,K)
ENDDO
ENDDO
ENDDO
!
IF(MULT .EQ. 5) RETURN
!---------------------------------------------------------------------
L=6
IF(I23 .EQ. 3) MZNI=MZ(L)
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AW6(I,J,K)=0.5*( AW5(2*I-2,2*J-1,2*K-2)+AW5(2*I-2,2*J-2,2*K-2)+AW5(2*I-2,2*J-1,2*K-1)+AW5(2*I-2,2*
J-2,2*K-1))
AE6(I,J,K)=0.5*( AE5(2*I-1,2*J-1,2*K-2)+AE5(2*I-1,2*J-2,2*K-2)+AE5(2*I-1,2*J-1,2*K-1)+AE5(2*I-1,2*J-2,
2*K-1))
AS6(I,J,K)=0.5*( AS5(2*I-1,2*J-2,2*K-2)+AS5(2*I-2,2*J-2,2*K-2)+AS5(2*I-1,2*J-2,2*K-1)+AS5(2*I-2,2*J-2,
2*K-1))
AN6(I,J,K)=0.5*( AN5(2*I-1,2*J-1,2*K-2)+AN5(2*I-2,2*J-1,2*K-2)+AN5(2*I-1,2*J-1,2*K-1)+AN5(2*I-2,2*J-1
,2*K-1))
AT6(I,J,K)=0.5*( AT5(2*I-1,2*J-2,2*K-1)+AT5(2*I-2,2*J-2,2*K-1)+AT5(2*I-1,2*J-1,2*K-1)+AT5(2*I-2,2*J-1,
2*K-1))
AB6(I,J,K)=0.5*( AB5(2*I-1,2*J-2,2*K-2)+AB5(2*I-2,2*J-2,2*K-2)+AB5(2*I-1,2*J-1,2*K-2)+AB5(2*I-2,2*J-1,
2*K-2))
ENDDO
ENDDO
ENDDO
```

```
!
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)
AP6(I,J,K)=AE6(I,J,K)+AN6(I,J,K)+AT6(I,J,K)+AW6(I,J,K)+AS6(I,J,K)+AB6(I,J,K)
ENDDO
ENDDO
ENDDO
!
RETURN
END

!**********************************************************************
SUBROUTINE MU_GESS
!**********************************************************************
USE PARAMETERS
USE IGRID
USE MUGRD
USE CALC1
USE M2APS
USE M3APS
USE M4APS
USE M5APS
USE M6APS
USE MPIVAR
IMPLICIT NONE
INTEGER :: L,I,J,K
!**********************************************************************
L=2
DO 120 K=2,MZ(L)
DO 120 J=2,MY(L)
DO 120 I=2,MX(L)
120     SS2(I,J,K)=SS(2*I-1,2*J-1,2*K-1)+SS(2*I-1,2*J-2,2*K-1)&
&               +SS(2*I-2,2*J-1,2*K-1)+SS(2*I-2,2*J-2,2*K-1)&
&               +SS(2*I-1,2*J-1,2*K-2)+SS(2*I-1,2*J-2,2*K-2)&
&               +SS(2*I-2,2*J-1,2*K-2)+SS(2*I-2,2*J-2,2*K-2)
!
IF(MULT .EQ. 2) RETURN
!
L=3
DO 130 K=2,MZ(L)
DO 130 J=2,MY(L)
DO 130 I=2,MX(L)
130     SS3(I,J,K)=SS2(2*I-1,2*J-1,2*K-1)+SS2(2*I-1,2*J-2,2*K-1)&
&               +SS2(2*I-2,2*J-1,2*K-1)+SS2(2*I-2,2*J-2,2*K-1)&
&               +SS2(2*I-1,2*J-1,2*K-2)+SS2(2*I-1,2*J-2,2*K-2)&
&               +SS2(2*I-2,2*J-1,2*K-2)+SS2(2*I-2,2*J-2,2*K-2)
!
IF(MULT .EQ. 3) RETURN
!
L=4
DO 140 K=2,MZ(L)
DO 140 J=2,MY(L)
```

```
DO 140 I=2,MX(L)
140     SS4(I,J,K)=SS3(2*I-1,2*J-1,2*K-1)+SS3(2*I-1,2*J-2,2*K-1)&
&               +SS3(2*I-2,2*J-1,2*K-1)+SS3(2*I-2,2*J-2,2*K-1)&
&               +SS3(2*I-1,2*J-1,2*K-2)+SS3(2*I-1,2*J-2,2*K-2)&
&               +SS3(2*I-2,2*J-1,2*K-2)+SS3(2*I-2,2*J-2,2*K-2)
!
IF(MULT .EQ. 4) RETURN
!
L=5
DO 150 K=2,MZ(L)
DO 150 J=2,MY(L)
DO 150 I=2,MX(L)
150     SS5(I,J,K)=SS4(2*I-1,2*J-1,2*K-1)+SS4(2*I-1,2*J-2,2*K-1)&
&               +SS4(2*I-2,2*J-1,2*K-1)+SS4(2*I-2,2*J-2,2*K-1)&
&               +SS4(2*I-1,2*J-1,2*K-2)+SS4(2*I-1,2*J-2,2*K-2)&
&               +SS4(2*I-2,2*J-1,2*K-2)+SS4(2*I-2,2*J-2,2*K-2)
!
IF(MULT .EQ. 5) RETURN
!
L=6
DO 160 K=2,MZ(L)
DO 160 J=2,MY(L)
DO 160 I=2,MX(L)
160     SS6(I,J,K)=SS5(2*I-1,2*J-1,2*K-1)+SS5(2*I-1,2*J-2,2*K-1)&
&               +SS5(2*I-2,2*J-1,2*K-1)+SS5(2*I-2,2*J-2,2*K-1)&
&               +SS5(2*I-1,2*J-1,2*K-2)+SS5(2*I-1,2*J-2,2*K-2)&
&               +SS5(2*I-2,2*J-1,2*K-2)+SS5(2*I-2,2*J-2,2*K-2)
!
RETURN
END

!***********************   MULTI GRID ***********************************
!**********************************************************************
SUBROUTINE MU_PRCC(I23,IEND,JEND,KEND,L)
!**********************************************************************
USE PARAMETERS
USE CALC1
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,IST=2,JST=2,KST=2,NT
INTEGER, INTENT(IN)::I23,IEND,JEND,KEND,L
REAL (KIND=8),DIMENSION(ID,JD,KD):: C
!**********************************************************************
DO 10 K=MZS(L),MZE(L)
DO 10 J=MYS(L),MYE(L)
DO 10 I=MXS(L),MXE(L)
10      C(I,J,K)=CC(I,J,K)

IIST=MXS(L)
JJST=MYS(L)
KKST=MZS(L)
IIEND=MXE(L)
JJEND=MYE(L)
```

```
KKEND=MZE(L)

CALL RELOAD

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(C(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(C(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(C(1,1,1),SIZE(C),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!------------------------
!
DO K=MZS(L),MZE(L)
DO J=MYS(L),MYE(L)

IF (PIDX.EQ.0) THEN
CC(2*IST-3,2*J-1,2*K-1)=0.
CC(2*IST-3,2*J-2,2*K-1)=0.
CC(2*IST-3,2*J-1,2*K-2)=0.
CC(2*IST-3,2*J-2,2*K-2)=0.
ENDIF

IF (PIDX.EQ.PROCSX-1) THEN
CC(2*IEND ,2*J-1,2*K-1)=0.
CC(2*IEND ,2*J-2,2*K-1)=0.
CC(2*IEND ,2*J-1,2*K-2)=0.
CC(2*IEND ,2*J-2,2*K-2)=0.
ENDIF

ENDDO
ENDDO
!
DO K=MZS(L),MZE(L)
DO I=MXS(L),MXE(L)

IF (PIDY.EQ.0) THEN
CC(2*I-1,2*JST-3,2*K-1)=0.
CC(2*I-2,2*JST-3,2*K-1)=0.
CC(2*I-1,2*JST-3,2*K-2)=0.
CC(2*I-2,2*JST-3,2*K-2)=0.
ENDIF
!
IF (PIDY.EQ.PROCSY-1) THEN
```

```
CC(2*I-1,2*JEND ,2*K-1)=0.
CC(2*I-2,2*JEND ,2*K-1)=0.
CC(2*I-1,2*JEND ,2*K-2)=0.
CC(2*I-2,2*JEND ,2*K-2)=0.
ENDIF

ENDDO
ENDDO
!
IF(I23 .EQ. 3) THEN
DO J=MYS(L),MYE(L)
DO I=MXS(L),MXE(L)

IF (PIDZ.EQ.0) THEN
CC(2*I-1,2*J-1,2*KST-3)=0.
CC(2*I-2,2*J-1,2*KST-3)=0.
CC(2*I-1,2*J-2,2*KST-3)=0.
CC(2*I-2,2*J-2,2*KST-3)=0.
ENDIF
!
IF (PIDZ.EQ.PROCSZ-1) THEN
CC(2*I-1,2*J-1 ,2*KEND)=0.
CC(2*I-2,2*J-1 ,2*KEND)=0.
CC(2*I-1,2*J-2 ,2*KEND)=0.
CC(2*I-2,2*J-2 ,2*KEND)=0.
ENDIF

ENDDO
ENDDO

ENDIF

!
DO 100 K= KST,KEND
DO 100 J= JST,JEND
DO 100 I= IST,IEND
CC(2*I-1,2*J-1,2*K-1)=C(I,J,K)
CC(2*I-2,2*J-1,2*K-1)=C(I,J,K)
CC(2*I-1,2*J-2,2*K-1)=C(I,J,K)
CC(2*I-2,2*J-2,2*K-1)=C(I,J,K)
CC(2*I-1,2*J-1,2*K-2)=C(I,J,K)
CC(2*I-2,2*J-1,2*K-2)=C(I,J,K)
CC(2*I-1,2*J-2,2*K-2)=C(I,J,K)
100     CC(2*I-2,2*J-2,2*K-2)=C(I,J,K)

!----------
IIST =MXS(L)
JJST =MYS(L)
KKST =MZS(L)
IIEND=MXE(L)
JJEND=MYE(L)
KKEND=MZE(L)
```

```
CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(CC(MXS(L):MXE(L),MYS(L):MYE(L),MZS(L):MZE(L)),SPANX*SPANY*SPANZ,MPI_DOUBL
E_PRECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(CC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(CC(1,1,1),SIZE(CC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
ENDIF
!------------
RETURN

END

!**********************************************************************
SUBROUTINE N_GET
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE IGRID
USE GRID1
USE GRIDB
USE LEVFN
USE FNXYZ
USE APANS
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT
REAL (KIND=8) :: DET,FX1,FY1,FZZ1
!**********************************************************************
CALL FUVW_GET

DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND
DO 100 I=IIST,IIEND
IF(I.EQ.1) THEN
FX1=(F(I+1,J,K)-F(I,J,K))/XD(I+1)
ELSEIF(I.EQ.NI) THEN
FX1=(F(I,J,K)-F(I-1,J,K))/XD(I)
ELSE
FX1=(FU(I+1,J,K)-FU(I,J,K))/XC(I)
ENDIF
```

```
IF(J.EQ.1) THEN
FY1=(F(I,J+1,K)-F(I,J,K))/YD(J+1)
ELSEIF(J.EQ.NJ) THEN
FY1=(F(I,J,K)-F(I,J-1,K))/YD(J)
ELSE
FY1=(FV(I,J+1,K)-FV(I,J,K))/YC(J)
ENDIF

IF(I23.EQ.2) GOTO 105
IF(K.EQ.1) THEN
FZZ1=(F(I,J,K+1)-F(I,J,K))/ZD(K+1)
ELSEIF(K.EQ.NK) THEN
FZZ1=(F(I,J,K)-F(I,J,K-1))/ZD(K)
ELSE
FZZ1=(FW(I,J,K+1)-FW(I,J,K))/ZC(K)
ENDIF

105     DET=AMAX1(SQRT(FX1**2+FY1**2+FZZ1**2),1.E-10)
FNX(I,J,K)=FX1/DET
FNY(I,J,K)=FY1/DET
FNZ(I,J,K)=FZZ1/DET
100     CONTINUE

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(FNX(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(FNX(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(FNX(1,1,1),SIZE(FNX),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(FNY(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(FNY(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF
```

```
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(FNY(1,1,1),SIZE(FNY),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!----------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(FNZ(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(FNZ(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(FNZ(1,1,1),SIZE(FNZ),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
!
SUBROUTINE OTHERS
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE QINTB
USE LEVFN
USE ROGAM
USE QMSOR
USE RESLT
USE APANS
USE INTXN
USE INTXY
USE UVTPP
USE CALC1
USE MPIVAR
IMPLICIT NONE
LOGICAL T_2PH,T_1PH
INTEGER :: IS,I,J,K,IND
INTEGER, INTENT(IN) :: IND0,I0,J0,K0
REAL (KIND=8) :: POINT
REAL (KIND=8),DIMENSION(100) :: QMIC1,QMIC2,FAC
!**********************************************************************
!
ENTRY INTP_GET(POINT,IND0,I0,J0,K0)
```

```
!**********************************************************************
POINT=0.
IS=1
IF(IND0 .LT. 0) IS=-1
IND=IABS(IND0)

IF(IND .EQ. 1) THEN
DO 110 I=1,NIM
IF(F(I,J0,K0)*F(I+1,J0,K0) .GT. 0.) GOTO 110
POINT=XP(I)-F(I,J0,K0)/(F(I+1,J0,K0)-F(I,J0,K0))*XD(I+1)
IF(IS.EQ.1) GOTO 115
110     CONTINUE
115     CONTINUE

ELSEIF(IND .EQ. 2) THEN
DO 120 J=1,NJM
IF(F(I0,J,K0)*F(I0,J+1,K0) .GT. 0.) GOTO 120
POINT=YP(J)-F(I0,J,K0)/(F(I0,J+1,K0)-F(I0,J,K0))*YD(J+1)
IF(IS.EQ.1) GOTO 125
120     CONTINUE
125     CONTINUE

ELSE
DO 130 K=1,NKM
IF(F(I0,J0,K)*F(I0,J0,K+1) .GT. 0.) GOTO 130
POINT=ZP(K)-F(I0,J0,K)/(F(I0,J0,K+1)-F(I0,J0,K))*ZD(K+1)
IF(IS.EQ.1) GOTO 135
130     CONTINUE
135     CONTINUE
ENDIF

RETURN
END

!**********************************************************************
SUBROUTINE OTHERS_SUB
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE UVTBC
USE LVFBC
USE QINTB
USE LEVFN
USE ROGAM
USE QMSOR
USE RESLT
```

```
USE APANS
USE INTXN
USE INTXY
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: N,I,J,K,IBUB1
REAL (KIND=8) ::
AR1,COSB,COTB,DDF1,DXE,DXW,DYN,DYS,QMICT,SINB,FDS,ASUM,AREA,QMIC1(100),QMIC2(100)
,FAC(100)
REAL (KIND=8) :: TTE,TTN,TTS,TTW,VFG1
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
!***********************************************************************
ENTRY QM_GET
!***********************************************************************
QMIC=0.
IF(ICONT.NE.1) RETURN
VFG1=VFG*HFGI/TK2
DO 100 N=1,NBUB(1)
QMIC1(N)=0.
100     QMIC2(N)=0.

IF(IBCW.NE.IWAL) GOTO 119
DO 110 K=2,NKM
DO 110 J=2,NJM
QBW(J,K)=0.
QMW(J,K)=0.
IF(S(2,J,K).LT.0.) GOTO 110
IBUB1=IBUB(2,J,K,1)
IF(IBUB1.LT.1) GOTO 110
COSB=COSBW(J,K)
DDF1=DDF(2,J,K)
IF(COSB.LE.0.) GOTO 110
SINB=AMAX1(1.E-8, SQRT(1.-COSB**2))
COTB=COSB/SINB
QMICT=QMIC0*COTB*TBCW

FDS=(F(2,J,K)/SINB+0.5*XD(2)*COTB)/DDF1
IF(ABS(FDS).LT.1.) QBW(J,K)=QMICT*0.5*(1.+COS(PI*FDS))/DDF1

FDS=((F(2,J,K)+DDF1)/SINB+DDF1)/DDF1
IF(ABS(FDS).LT.1.) QMW(J,K)=QMICT*0.5*(1.+COS(PI*FDS))/DDF1

AR1=RU(2)*YC(J)*ZC(K)
QMIC1(IBUB1)=QMIC1(IBUB1)+QBW(J,K)*AR1
QMIC2(IBUB1)=QMIC2(IBUB1)+QMW(J,K)*AR1
110     CONTINUE

DO 114 N=1,NBUB(1)
QMIC=QMIC+QMIC1(N)
```

```
FAC(N)=QMIC1(N)/AMAX1(1.E-8,QMIC2(N))
FAC(N)=AMIN1(FAC(N),5.)
QMIC1(N)=0.
114     QMIC2(N)=0.

DO 116 K=2,NKM
DO 116 J=2,NJM
IBUB1=IBUB(2,J,K,1)
IF(IBUB1.LT.1) GOTO 116
QMW(J,K)=FAC(IBUB1)*QMW(J,K)
VM(2,J,K)=VM(2,J,K)+VFG1*QMW(J,K)*RU(2)*YC(J)*ZC(K)
116     CONTINUE

119     IF(IBCE.NE.IWAL) GOTO 129
!---------------------------------------
129     IF(IBCS.NE.IWAL) GOTO 139
DO 130 K=2,NKM
DO 130 I=2,NIM
QBS(I,K)=0.
QMS(I,K)=0.
IF(S(I,2,K).LT.0.) GOTO 130
IBUB1=IBUB(I,2,K,1)
IF(IBUB1.LT.1) GOTO 130
COSB=COSBS(I,K)
DDF1=DDF(I,2,K)
IF(COSB.LE.0.) GOTO 130
SINB=AMAX1(1.E-8, SQRT(1.-COSB**2))
COTB=COSB/SINB
QMICT=QMIC0*COTB*TBCS

FDS=(F(I,2,K)/SINB+0.5*YD(2)*COTB)/DDF1
IF(ABS(FDS).LT.1.) QBS(I,K)=QMICT*0.5*(1.+COS(PI*FDS))/DDF1

FDS=((F(I,2,K)+DDF1)/SINB+DDF1)/DDF1
IF(ABS(FDS).LT.1.) QMS(I,K)=QMICT*0.5*(1.+COS(PI*FDS))/DDF1

AR1=RP(I)*XC(I)*ZC(K)
QMIC1(IBUB1)=QMIC1(IBUB1)+QBS(I,K)*AR1
QMIC2(IBUB1)=QMIC2(IBUB1)+QMS(I,K)*AR1
130     CONTINUE

DO 134 N=1,NBUB(1)
QMIC=QMIC+QMIC1(N)
FAC(N)=QMIC1(N)/AMAX1(1.E-8,QMIC2(N))
FAC(N)=AMIN1(FAC(N),5.)
QMIC1(N)=0.
134     QMIC2(N)=0.

DO 136 K=2,NKM
DO 136 I=2,NIM
IBUB1=IBUB(I,2,K,1)
IF(IBUB1.LT.1) GOTO 136
QMS(I,K)=FAC(IBUB1)*QMS(I,K)
```

```
VM(I,2,K)=VM(I,2,K)+VFG1*QMS(I,K)*RP(I)*XC(I)*ZC(K)
136     CONTINUE

139     IF(IBCN.NE.IWAL) GOTO 149
149     IF(IBCB.NE.IWAL) GOTO 159
159     IF(IBCT.NE.IWAL) RETURN

RETURN
!***********************************************************************
ENTRY QW_GET
!***********************************************************************
ASUM=0.
QWAV=0.
QMAV=0.

IF(IBCW.NE.IWAL) GOTO 315
DO 310 K=2,NKM
DO 310 J=2,NJM
DXE=XD(2)
TTE=T(2,J,K)
CALL DX_TT(1,2,J,K,DXE,TTE,T_FE,T_SE)
QWW(J,K)=TK2/TKP(1,J,K)*(T(1,J,K)-TTE)/DXE+QBW(J,K)
AREA=RU(2)*YC(J)*ZC(K)
ASUM=ASUM+AREA
QMAV=QMAV+QBW(J,K)*AREA
310     QWAV=QWAV+QWW(J,K)*AREA

315     IF(IBCE.NE.IWAL) GOTO 325
DO 320 K=2,NKM
DO 320 J=2,NJM
DXW=XD(NI)
TTW=T(NIM,J,K)
CALL DX_TT(NI,NIM,J,K,DXW,TTW,T_FW,T_SW)
QWE(J,K)=TK2/TKP(NI,J,K)*(T(NI,J,K)-TTW)/DXW+QBE(J,K)
AREA=RU(NI)*YC(J)*ZC(K)
ASUM=ASUM+AREA
QMAV=QMAV+QBE(J,K)*AREA
320     QWAV=QWAV+QWE(J,K)*AREA
!
325     IF(IBCS.NE.IWAL) GOTO 335
DO 330 K=2,NKM
DO 330 I=2,NIM
DYN=YD(2)
TTN=T(I,2,K)
CALL DY_TT(I,1,2,K,DYN,TTN,T_FN,T_SN)
QWS(I,K)=TK2/TKP(I,1,K)*(T(I,1,K)-TTN)/DYN+QBS(I,K)
AREA=RP(I)*XC(I)*ZC(K)
ASUM=ASUM+AREA
QMAV=QMAV+QBS(I,K)*AREA
330     QWAV=QWAV+QWS(I,K)*AREA

335     IF(IBCN.NE.IWAL) GOTO 345
DO 340 K=2,NKM
```

```
DO 340 I=2,NIM
DYS=YD(NJ)
TTS=T(I,NJM,K)
CALL DY_TT(I,NJ,NJM,K,DYS,TTS,T_FS,T_SS)
QWN(I,K)=TK2/TKP(I,NJ,K)*(T(I,NJ,K)-TTS)/DYS+QBN(I,K)
AREA=RP(I)*XC(I)*ZC(K)
ASUM=ASUM+AREA
QMAV=QMAV+QBN(I,K)*AREA
340     QWAV=QWAV+QWN(I,K)*AREA

345     IF(IBCB.NE.IWAL) GOTO 355
355     IF(IBCT.NE.IWAL) GOTO 365
365     CONTINUE

QWAV=QWAV/ARO
QMAV=QMAV/ARO

RETURN
!**********************************************************************
ENTRY QI_GET
!**********************************************************************
QIAV=0.
QIA1=0.
IF(ABS(HFGI/TK2).LT.1.E-8) RETURN

DO 410 K=2,NKM
DO 410 J=2,NJM
DO 410 I=2,NI
IF(T_1PH(F(I,J,K),F(I-1,J,K))) GOTO 410
QIAV=QIAV+AMX(I,J,K)*RU(I)*YC(J)*ZC(K)*SIGN1(F(I,J,K)-F(I-1,J,K))
410     CONTINUE

DO 420 K=2,NKM
DO 420 J=2,NJ
DO 420 I=2,NIM
IF(T_1PH(F(I,J,K),F(I,J-1,K))) GOTO 420
QIAV=QIAV+AMY(I,J,K)*RP(I)*XC(I)*ZC(K)*SIGN1(F(I,J,K)-F(I,J-1,K))
420     CONTINUE

IF(I23.EQ.2) GOTO 435
DO 430 K=2,NK
DO 430 J=2,NJM
DO 430 I=2,NIM
IF(T_1PH(F(I,J,K),F(I,J,K-1))) GOTO 430
QIAV=QIAV+AMZ(I,J,K)*XC(I)*YC(J)*SIGN1(F(I,J,K)-F(I,J,K-1))
430     CONTINUE

435     CONTINUE

DO 440 K=2,NKM
DO 440 J=2,NJM
DO 440 I=2,NIM
440     QIA1=QIA1+VM(I,J,K)
```

```
QIAV=QIAV*TK2/HFGI+QMIC
QIA1=QIA1*TK2/HFGI/VFG
QIAV=QIAV/ARO
QIA1=QIA1/ARO
RETURN
END
!************************************************************************
SUBROUTINE P_GEAS
!************************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT
REAL (KIND=8) :: HBW,HBS,HBB,FLUX1,FLUX2,FLUX3
LOGICAL T_2PH,T_1PH
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
!************************************************************************
CALL SHRINK
CALL RELOAD

DO 10 K=2,NKM
DO 10 J=2,NJM
DO 10 I=2,NIM
FLUX1=(RU(I)*U(I,J,K)-RU(I+1)*U(I+1,J,K))*YC(J)*ZC(K)
FLUX2=(V(I,J,K)-V(I,J+1,K))*RP(I)*XC(I)*ZC(K)
IF(I23.EQ.3) FLUX3=(W(I,J,K)-W(I,J,K+1))*XC(I)*YC(J)
PSOR(I,J,K)=(VM(I,J,K)+FLUX1+FLUX2+FLUX3)/DT
10      CONTINUE
!-------------------------------------------X-DIRECTION
DO 110 K=KKST,KKEND
DO 110 J=JJST,JJEND
DO 110 I=IIST,IIEND
HBW=1.
IF(SU(I,J,K).LE.0.) HBW=1.E-8
AE(I,J,K)=HBW*RU(I+1)*YC(J)*ZC(K)/RHOU(I+1,J,K)/XD(I+1)
```

```
110     AW(I,J,K)=HBW*RU(I)*YC(J)*ZC(K)/RHOU(I,J,K)/XD(I)
!-----------------------------------------Y-DIRECTION
DO 120 K=KKST,KKEND
DO 120 J=JJST,JJEND
DO 120 I=IIST,IIEND
HBS=1.
IF(SV(I,J,K).LE.0.) HBS=1.E-8
AN(I,J,K)=HBS*RP(I)*XC(I)*ZC(K)/RHOV(I,J+1,K)/YD(J+1)
120     AS(I,J,K)=HBS*RP(I)*XC(I)*ZC(K)/RHOV(I,J,K)/YD(J)
!-----------------------------------------Y-DIRECTION
IF(I23.EQ.2) GOTO 135
DO 130 K=KKST,KKEND
DO 130 J=JJST,JJEND
DO 130 I=IIST,IIEND
HBB=1.
IF(SW(I,J,K).LE.0.) HBB=1.E-8
AT(I,J,K)=HBB*XC(I)*YC(J)/RHOW(I,J,K+1)/ZD(K+1)
130     AB(I,J,K)=HBB*XC(I)*YC(J)/RHOW(I,J,K)/ZD(K)
135     CONTINUE
!-----------------------------------------BC
DO 210 K=KKST,KKEND
DO 210 J=JJST,JJEND
IF(IBCW.NE.IOUT) AW( 2,J,K)=0.
IF(IBCE.NE.IOUT) THEN
AW(NI,J,K)=0.
AE(NIM,J,K)=0.
ENDIF
210     CONTINUE

DO 220 K=KKST,KKEND
DO 220 I=IIST,IIEND
IF(IBCS.NE.IOUT) AS(I, 2,K)=0.
IF(IBCN.NE.IOUT) THEN
AS(I,NJ,K)=0.
AN(I,NJM,K)=0.
ENDIF
220     CONTINUE

DO 230 J=JJST,JJEND
DO 230 I=IIST,IIEND
IF(IBCB.NE.IOUT) AB(I,J, 2)=0.
IF(IBCT.NE.IOUT) THEN
AB(I,J,NK)=0.
AT(I,J,NKM)=0.
ENDIF
230     CONTINUE
!-----------------------------------------
DO 300 K=KKST,KKEND
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND
AP(I,J,K)=AE(I,J,K)+AW(I,J,K)+AN(I,J,K)+AS(I,J,K)+AT(I,J,K)+AB(I,J,K)
300     APC(I,J,K)=AP(I,J,K)
!-----------------------------------------
```

```
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(AE(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AE(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AE(1,1,1),SIZE(AE),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!-------------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(AW(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AW(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AW(1,1,1),SIZE(AW),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!--------------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(AN(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AN(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AN(1,1,1),SIZE(AN),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!--------------------------
IF (TASKID.NE.0) THEN
```

```
CALL
MPI_SEND(AS(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AS(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AS(1,1,1),SIZE(AS),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!------------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(AT(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AT(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AT(1,1,1),SIZE(AT),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!-------------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(AB(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AB(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AB(1,1,1),SIZE(AB),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!---------------------------
IF (TASKID.NE.0) THEN
```

```
CALL
MPI_SEND(AP(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(AP(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(AP(1,1,1),SIZE(AP),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!------------------------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(APC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(APC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(
NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(APC(1,1,1),SIZE(APC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE PCGITER
!**********************************************************************
USE PARAMETERS
USE IGRID
USE SPERR
USE APANS
USE UVTPP
USE CALC1
USE RELXS
USE WORK1
USE CGSSS
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,ICGS=1
INTEGER, INTENT(IN) :: N,ITER
REAL (KIND=8) :: BETA0=0.,CC_A_FC,FCM,FFM,REL1,RELA,SSM
```

```
REAL (KIND=8) :: DUMMY,RCC_A_FC,RFC_A_FC
REAL (KIND=8) :: RFCC,RFF,RFC_SS,RFCM,RFFM,RSS,RSSM
INTEGER :: NT

!***********************************************************************
ENTRY ITERM1(N,ITER)
!***********************************************************************
CALL RELOAD
IF(ICGS.EQ.0) GOTO 290
IF(ITER.EQ.0) BETA0=0.
IF(ITER.NE.0) THEN
CC_A_FC=0.

DO K= KKST,KKEND
DO J= JJST,JJEND
DO I= IIST,IIEND
CC_A_FC=CC_A_FC+CC(I,J,K)*A_FC(I,J,K)
ENDDO
ENDDO
ENDDO
CALL
MPI_ALLREDUCE(CC_A_FC,RCC_A_FC,1,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORL
D,IERR)
CC_A_FC=RCC_A_FC

BETA0=-CC_A_FC/FC_A_FC
ENDIF

RELX2(N)=BETA0

290     FC_SS=0.

DO K=KKST,KKEND
DO J=JJST,JJEND
DO I=IIST,IIEND

IF (N.EQ.1) FF(I,J,K,N)= U(I,J,K)
IF (N.EQ.2) FF(I,J,K,N)= V(I,J,K)
IF (N.EQ.3) FF(I,J,K,N)= W(I,J,K)
IF (N.EQ.4) FF(I,J,K,N)= T(I,J,K)
IF (N.EQ.5) FF(I,J,K,N)= P(I,J,K)

BS(I,J,K)=SS(I,J,K)
FCC(I,J,K)=CC(I,J,K)+BETA0*FCC(I,J,K)
FF(I,J,K,N)=FF(I,J,K,N)+FCC(I,J,K)
FC_SS=FC_SS+FCC(I,J,K)*SS(I,J,K)

IF (N.EQ.1) U(I,J,K)= FF(I,J,K,N)
IF (N.EQ.2) V(I,J,K)= FF(I,J,K,N)
IF (N.EQ.3) W(I,J,K)= FF(I,J,K,N)
IF (N.EQ.4) T(I,J,K)= FF(I,J,K,N)
IF (N.EQ.5) P(I,J,K)= FF(I,J,K,N)
ENDDO
```

```
ENDDO
ENDDO

CALL
MPI_ALLREDUCE(FC_SS,RFC_SS,1,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,IER
R)
FC_SS=RFC_SS

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(BS(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,3,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(BS(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,3,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BCAST(BS(1,1,1),SIZE(BS),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(FCC(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PREC
ISION,0,3,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(FCC(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(
NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,3,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BCAST(FCC(1,1,1),SIZE(FCC),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
!
IF (TASKID.NE.0) THEN
IF (N.EQ.1) CALL
MPI_SEND(U(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,1,MPI_COMM_WORLD,IERR)
IF (N.EQ.2) CALL
MPI_SEND(V(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,2,MPI_COMM_WORLD,IERR)
IF (N.EQ.3) CALL
MPI_SEND(W(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECIS
ION,0,3,MPI_COMM_WORLD,IERR)
```

```
IF (N.EQ.4) CALL
MPI_SEND(T(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,4,MPI_COMM_WORLD,IERR)
IF (N.EQ.5) CALL
MPI_SEND(P(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,5,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
IF (N.EQ.1) CALL
MPI_RECV(U(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,1,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.2) CALL
MPI_RECV(V(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,2,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.3) CALL
MPI_RECV(W(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT)
),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,3,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.4) CALL
MPI_RECV(T(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,4,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.5) CALL
MPI_RECV(P(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,5,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
IF (N.EQ.1) CALL
MPI_BCAST(U(1,1,1),SIZE(U),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.2) CALL
MPI_BCAST(V(1,1,1),SIZE(V),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.3) CALL
MPI_BCAST(W(1,1,1),SIZE(W),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.4) CALL
MPI_BCAST(T(1,1,1),SIZE(T),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.5) CALL
MPI_BCAST(P(1,1,1),SIZE(P),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
!**************************************************************************
ENTRY ITERM2(N,ITER)
!**************************************************************************
FC_A_FC=0.
DO 400 K=KKST,KKEND
DO 400 J=JJST,JJEND
DO 400 I=IIST,IIEND
A_FC(I,J,K)=BS(I,J,K)-SS(I,J,K)
400     FC_A_FC=FC_A_FC+FCC(I,J,K)*A_FC(I,J,K)
CALL
MPI_ALLREDUCE(FC_A_FC,RFC_A_FC,1,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORL
D,IERR)
```

```
FC_A_FC=RFC_A_FC
FC_A_FC=SIGN(AMAX1(ABS(FC_A_FC),1.E-99),FC_A_FC)

RELA=FC_SS/FC_A_FC
RELAX(N)=RELA
BRELX(N)=RELAX(N)

FFM=0.
FCM=0.
SSM=0.
RELA=RELAX(N)
REL1=1.-RELA

DO 430 K=KKST,KKEND
DO 430 J=JJST,JJEND
DO 430 I=IIST,IIEND

IF (N.EQ.1) FF(I,J,K,N)= U(I,J,K)
IF (N.EQ.2) FF(I,J,K,N)= V(I,J,K)
IF (N.EQ.3) FF(I,J,K,N)= W(I,J,K)
IF (N.EQ.4) FF(I,J,K,N)= T(I,J,K)
IF (N.EQ.5) FF(I,J,K,N)= P(I,J,K)

FF(I,J,K,N)=FF(I,J,K,N)-REL1*FCC(I,J,K)
SS(I,J,K)   =SS(I,J,K)   +REL1*A_FC(I,J,K)
SSM=AMAX1(SSM,ABS(SS(I,J,K)/AP(I,J,K)))
FFM=AMAX1(FFM,ABS(FF(I,J,K,N)))
FCM=AMAX1(FCM,ABS(FCC(I,J,K)))

IF (N.EQ.1) U(I,J,K)= FF(I,J,K,N)
IF (N.EQ.2) V(I,J,K)= FF(I,J,K,N)
IF (N.EQ.3) W(I,J,K)= FF(I,J,K,N)
IF (N.EQ.4) T(I,J,K)= FF(I,J,K,N)
IF (N.EQ.5) P(I,J,K)= FF(I,J,K,N)

430     CONTINUE

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(SS(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,3,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(SS(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,3,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BCAST(SS(1,1,1),SIZE(SS),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
```

```
IF (TASKID.NE.0) THEN
IF (N.EQ.1) CALL
MPI_SEND(U(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,1,MPI_COMM_WORLD,IERR)
IF (N.EQ.2) CALL
MPI_SEND(V(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,2,MPI_COMM_WORLD,IERR)
IF (N.EQ.3) CALL
MPI_SEND(W(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECIS
ION,0,3,MPI_COMM_WORLD,IERR)
IF (N.EQ.4) CALL
MPI_SEND(T(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,4,MPI_COMM_WORLD,IERR)
IF (N.EQ.5) CALL
MPI_SEND(P(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISI
ON,0,5,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
IF (N.EQ.1) CALL
MPI_RECV(U(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,1,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.2) CALL
MPI_RECV(V(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,2,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.3) CALL
MPI_RECV(W(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT)
),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,3,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.4) CALL
MPI_RECV(T(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,4,MPI_COMM_WORLD,STATUS,IERR)
IF (N.EQ.5) CALL
MPI_RECV(P(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT))
,SIZECC(NT),MPI_DOUBLE_PRECISION,NT,5,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
IF (N.EQ.1) CALL
MPI_BCAST(U(1,1,1),SIZE(U),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.2) CALL
MPI_BCAST(V(1,1,1),SIZE(V),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.3) CALL
MPI_BCAST(W(1,1,1),SIZE(W),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.4) CALL
MPI_BCAST(T(1,1,1),SIZE(T),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
IF (N.EQ.5) CALL
MPI_BCAST(P(1,1,1),SIZE(P),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

CALL
MPI_ALLREDUCE(SSM,RSSM,1,MPI_DOUBLE_PRECISION,MPI_MAX,MPI_COMM_WORLD,IERR)
```

```
CALL
MPI_ALLREDUCE(FFM,RFFM,1,MPI_DOUBLE_PRECISION,MPI_MAX,MPI_COMM_WORLD,IERR)
CALL
MPI_ALLREDUCE(FCM,RFCM,1,MPI_DOUBLE_PRECISION,MPI_MAX,MPI_COMM_WORLD,IERR)
SSM=RSSM
FFM=RFFM
FCM=RFCM
FFM=AMAX1(1.E-30,FFM)
FFMAX(N)=FFM
SSMAX(N)=SSM/FFM
FCMAX(N)=FCM/FFM

RETURN
END
!***********************************************************************
SUBROUTINE PROJT
!***********************************************************************
USE PARAMETERS
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE LEVFN
USE ROGAM
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT
!***********************************************************************
DO 110 K=2,NKM
DO 110 J=2,NJM
DO 110 I=2,NI
IF(SU(I,J,K).LE.0.) GOTO 110
U(I,J,K)=U(I,J,K)-DT*(P(I,J,K)-P(I-1,J,K))/RHOU(I,J,K)/XD(I)
110     CONTINUE

DO 120 K=2,NKM
DO 120 J=2,NJ
DO 120 I=2,NIM
IF(SV(I,J,K).LE.0.) GOTO 120
V(I,J,K)=V(I,J,K)-DT*(P(I,J,K)-P(I,J-1,K))/RHOV(I,J,K)/YD(J)
120     CONTINUE

IF(I23.EQ.2) RETURN
DO 130 K=2,NK
DO 130 J=2,NJM
DO 130 I=2,NIM
IF(SW(I,J,K).LE.0.) GOTO 130
W(I,J,K)=W(I,J,K)-DT*(P(I,J,K)-P(I,J,K-1))/RHOW(I,J,K)/ZD(K)
130     CONTINUE

RETURN
END
```

```
!***********************************************************************
SUBROUTINE QN_EXT(IPHASE)
!***********************************************************************
USE PARAMETERS
USE INDMO
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
USE ICHEK
USE CALC1
USE IJKMK
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,N,L,INC,NN,II,JJ,KK,III,JJJ,KKK,ITER,NCHB0
INTEGER, INTENT(IN) :: IPHASE
REAL (KIND=8) :: FMIN
LOGICAL T_2PH,T_1PH
!***********************************************************************
DO 1 K=K11,KNK
DO 1 J=1,NJ
DO 1 I=1,NI
AC(I,J,K)=0.
SS(I,J,K)=0.
IMK(I,J,K)=0
JMK(I,J,K)=0
KMK(I,J,K)=0
1       CONTINUE

DO 2 N=1,NCHB
I=IIB(N)
J=JJB(N)
K=KKB(N)
IMK(I,J,K)=1
JMK(I,J,K)=1
KMK(I,J,K)=1
2       CONTINUE
!
ITER=0
10      ITER=ITER+1
!-------------------------------------------   CHECK
NCHB0=NCHB
DO 100 N=1,NCHB0
I=IIB(N)
J=JJB(N)
K=KKB(N)
IF(MK0(I,J,K).GE.2) GOTO 100
```

```
DO 110 KK=MAX(K-1,K11),MIN(K+1,KNK)
DO 110 JJ=MAX(J-1,1),MIN(J+1,NJ)
DO 110 II=MAX(I-1,1),MIN(I+1,NI)
IF(KK.EQ.K .AND. JJ.EQ.J .AND. II.EQ.I) GOTO 110
IF(MK0(II,JJ,KK).LE.2) GOTO 110
NCHB=NCHB+1
IIB(NCHB)=II
JJB(NCHB)=JJ
KKB(NCHB)=KK
MK0(II,JJ,KK)=2
110     CONTINUE
100     CONTINUE
!
N=0
INC=1
DO 290 L=1,2
IF(L.EQ.2) INC=-1
210     N=N+INC
IF(N.GT.NCHB .OR. N.LT.1) GOTO 290
!
I=IIB(N)
J=JJB(N)
K=KKB(N)
IF(MK0(I,J,K).LE.1) GOTO 210
CALL QN_SUB(I,J,K,IPHASE)
GOTO 210
290     CONTINUE
!
IF(ITER.LT.10) GOTO 10
!
DO 300 N=1,NCHB
I=IIB(N)
J=JJB(N)
K=KKB(N)
IF(MK0(I,J,K).NE.0) GOTO 300
DO 310 KK=MAX(K-4,K11),MIN(K+4,KNK)
DO 310 JJ=MAX(J-4,1),MIN(J+4,NJ)
DO 310 II=MAX(I-4,1),MIN(I+4,NI)
IF(MK0(II,JJ,KK).LE.1) GOTO 310
9001    FORMAT(A,I6,4(I4),A,3(I2),A,4(I2))
NN=0
312     NN=NN+1
FMIN=1.E30
DO 314 KKK=MAX(KK-NN,K11),MIN(KK+NN,KNK)
DO 314 JJJ=MAX(JJ-NN,1),MIN(JJ+NN,NJ)
DO 314 III=MAX(II-NN,1),MIN(II+NN,NI)
IF(MK0(III,JJJ,KKK).GT.1) GOTO 314
IF(ABS(F(II,JJ,KKK)-F(III,JJJ,KKK)).GT.FMIN) GOTO 314
QN(II,JJ,KK,IPHASE)=QN(III,JJJ,KKK,IPHASE)
FMIN=ABS(F(II,JJ,KK)-F(III,JJJ,KKK))
MK0(II,JJ,KK)=1
314     CONTINUE
IF(MK0(II,JJ,KK).LE.1) GOTO 310
```

```
IF(NN.EQ.5) THEN
WRITE(I69,*) ' QN_EXT ERR',ITIME,I,J,K,II,JJ,KK
STOP
ENDIF
GOTO 312
310     CONTINUE
300     CONTINUE
!
RETURN
END

!***********************************************************************
SUBROUTINE QN_GET
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE ROGAM
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
USE RESLT
USE ICHEK
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L,L0,L9,KM1,KP1,IM1,IP1,JM1,JP1,III,JJJ,KKK
REAL (KIND=8) ::
DXN,DXS,DXW,DXE,DZB,DZT,DYN,DYS,DXE0,DXW0,DXN0,DXS0,DZB0,DZT0,DYN0,DYS0,DXE1,DX
W1,DYN1,DYS1,DZB1,DZT1
REAL (KIND=8) ::
HB0,HE0,HN0,HS0,HT0,HW0,TXP,TYP,TTB,TTE,TTN,TTP,TTS,TTW,TXE,TXW,TYN,TYS,TTT,TZB,TZP
,TZT
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT,T_FP,T_NOINTF
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
LOGICAL T_3W,T_3E,T_3S,T_3N,T_3B,T_3T
LOGICAL T_IW,T_IE,T_IS,T_IN,T_IB,T_IT,T_IP
!***********************************************************************
L0=1
L9=2
IF(IQ_VAP.EQ.0) L0=2
IF(IQ_LIQ.EQ.0) L9=1
```

```
DO 1000 L=L0,L9

NCHB=0
DO 100 K=K11,KNK
DO 100 J=1,NJ
DO 100 I=1,NI
QN(I,J,K,L)=0.
MK0(I,J,K)=3
IF(S(I,J,K).LE.0.) GOTO 100
T_FP=(L.EQ.1 .AND. F(I,J,K).GE.0.).OR.(L.EQ.2 .AND. F(I,J,K).LE.0.)
IF(IBODY.EQ.0 .AND. T_FP) GOTO 100

IM1=MAX(I-1, 1)
IP1=MIN(I+1,NI)
JM1=MAX(J-1, 1)
JP1=MIN(J+1,NJ)

T_FW=T_2PH(F(I,J,K),F(IM1,J,K))
T_FE=T_2PH(F(I,J,K),F(IP1,J,K))
T_FS=T_2PH(F(I,J,K),F(I,JM1,K))
T_FN=T_2PH(F(I,J,K),F(I,JP1,K))

T_NOINTF=.NOT. (T_FW .OR. T_FE .OR. T_FS .OR. T_FN)
IF(I23.EQ.2 .AND. T_NOINTF) GOTO 100
IF(I23.EQ.3) THEN
KM1=MAX(K-1, 1)
KP1=MIN(K+1,NK)
T_FB=T_2PH(F(I,J,K),F(I,J,KM1))
T_FT=T_2PH(F(I,J,K),F(I,J,KP1))
T_NOINTF=T_NOINTF .AND. (.NOT. (T_FB .OR. T_FT))
IF(T_NOINTF) GOTO 100
ENDIF

T_SW=T_2PH(S(I,J,K),S(IM1,J,K))
T_SE=T_2PH(S(I,J,K),S(IP1,J,K))
T_SS=T_2PH(S(I,J,K),S(I,JM1,K))
T_SN=T_2PH(S(I,J,K),S(I,JP1,K))

T_3W=T_FW .AND. T_SW
T_3E=T_FE .AND. T_SE
T_3S=T_FS .AND. T_SS
T_3N=T_FN .AND. T_SN

IF(T_3W) T_FW=(F(I,J,K)/(F(I,J,K)-F(IM1,J,K))).LT.(S(I,J,K)/(S(I,J,K)-S(IM1,J,K)))
IF(T_3E) T_FE=(F(I,J,K)/(F(I,J,K)-F(IP1,J,K))).LT.(S(I,J,K)/(S(I,J,K)-S(IP1,J,K)))
IF(T_3S) T_FS=(F(I,J,K)/(F(I,J,K)-F(I,JM1,K))).LT.(S(I,J,K)/(S(I,J,K)-S(I,JM1,K)))
IF(T_3N) T_FN=(F(I,J,K)/(F(I,J,K)-F(I,JP1,K))).LT.(S(I,J,K)/(S(I,J,K)-S(I,JP1,K)))
!
IF(T_3W) T_SW=.NOT. T_FW
IF(T_3E) T_SE=.NOT. T_FE
IF(T_3S) T_SS=.NOT. T_FS
IF(T_3N) T_SN=.NOT. T_FN
!
```

```
T_NOINTF=.NOT. (T_FW .OR. T_FE .OR. T_FS .OR. T_FN)
IF(I23.EQ.2 .AND. T_NOINTF) GOTO 100
!----------------------
IF(I23.EQ.3) THEN
T_SB=T_2PH(S(I,J,K),S(I,J,KM1))
T_ST=T_2PH(S(I,J,K),S(I,J,KP1))
T_3B=T_FB .AND. T_SB
T_3T=T_FT .AND. T_ST
IF(T_3B) T_FB=(F(I,J,K)/(F(I,J,K)-F(I,J,KM1))).LT.&
&                    (S(I,J,K)/(S(I,J,K)-S(I,J,KM1)))
IF(T_3T) T_FT=(F(I,J,K)/(F(I,J,K)-F(I,J,KP1))).LT.&
&                    (S(I,J,K)/(S(I,J,K)-S(I,J,KP1)))
IF(T_3B) T_SB=.NOT. T_FB
IF(T_3T) T_ST=.NOT. T_FT
!
T_NOINTF=T_NOINTF .AND. (.NOT. (T_FB .OR. T_FT))
IF(T_NOINTF) GOTO 100
ENDIF
!---------------------------
T_IW=T_3W .AND. T_FW
T_IE=T_3E .AND. T_FE
T_IS=T_3S .AND. T_FS
T_IN=T_3N .AND. T_FN

T_IP=.NOT. (T_IW .OR. T_IE .OR. T_IS .OR. T_IN)
IF(I23.EQ.2 .AND. T_FP .AND. T_IP) GOTO 100
IF(I23.EQ.3) THEN
T_IB=T_3B .AND. T_FB
T_IT=T_3T .AND. T_FT
T_IP=T_IP .AND. (.NOT. (T_IB .OR. T_IT))
IF(T_FP .AND. T_IP) GOTO 100
ENDIF

NCHB=NCHB+1
IIB(NCHB)=I
JJB(NCHB)=J
KKB(NCHB)=K
MK0(I,J,K)=0

III=MAX(I,2)
JJJ=MAX(J,2)
DXW0=XD(III)
DXE0=XD(IP1)
DYS0=YD(JJJ)
DYN0=YD(JP1)

DXW=DXW0
DXE=DXE0
DYS=DYS0
DYN=DYN0

IF(T_FW) HW0=F(I,J,K)/(F(I,J,K)-F(IM1,J,K))
IF(T_FE) HE0=F(I,J,K)/(F(I,J,K)-F(IP1,J,K))
```

```
IF(T_FS) HS0=F(I,J,K)/(F(I,J,K)-F(I,JM1,K))
IF(T_FN) HN0=F(I,J,K)/(F(I,J,K)-F(I,JP1,K))

IF(T_FW) DXW=DXW0*AMAX1(HEPS,HW0)
IF(T_FE) DXE=DXE0*AMAX1(HEPS,HE0)
IF(T_FS) DYS=DYS0*AMAX1(HEPS,HS0)
IF(T_FN) DYN=DYN0*AMAX1(HEPS,HN0)

TXP=0.
TYP=0.

IF(T_FP) THEN
IF(T_IW) DXW1=DXW0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(IM1,J,K)))
IF(T_IE) DXE1=DXE0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(IP1,J,K)))
IF(T_IS) DYS1=DYS0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,JM1,K)))
IF(T_IN) DYN1=DYN0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,JP1,K)))
IF(T_IW) TXW=(TINT-TWAL)/AMAX1(HEPS*DXW0,DXW1-DXW)
IF(T_IE) TXE=(TWAL-TINT)/AMAX1(HEPS*DXE0,DXE1-DXE)
IF(T_IS) TYS=(TINT-TWAL)/AMAX1(HEPS*DYS0,DYS1-DYS)
IF(T_IN) TYN=(TWAL-TINT)/AMAX1(HEPS*DYN0,DYN1-DYN)
IF(T_IW) TXP=TXW
IF(T_IE) TXP=TXE
IF(T_IW .AND. T_IE) TXP=0.5*(TXW+TXE)
IF(T_IS) TYP=TYS
IF(T_IN) TYP=TYN
IF(T_IS .AND. T_IN) TYP=0.5*(TYS+TYN)
GOTO 110
ENDIF

IF(T_SW) DXW=DXW0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(IM1,J,K)))
IF(T_SE) DXE=DXE0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(IP1,J,K)))
IF(T_SS) DYS=DYS0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,JM1,K)))
IF(T_SN) DYN=DYN0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,JP1,K)))

TTP=T(I,J,K)
TTW=T(IM1,J,K)
TTE=T(IP1,J,K)
TTS=T(I,JM1,K)
TTN=T(I,JP1,K)

IF(T_FW) TTW=TINT
IF(T_FE) TTE=TINT
IF(T_FS) TTS=TINT
IF(T_FN) TTN=TINT

TXW=(TTP-TTW)/DXW
TXE=(TTE-TTP)/DXE
TYS=(TTP-TTS)/DYS
TYN=(TTN-TTP)/DYN

IF(T_FW .AND. T_FE) THEN
TXP=0.5*(TXW+TXE)
ELSEIF(T_FW) THEN
```

```
IF(I.NE.NI .AND. HW0.LT.HEPS)&
&       TXW=(TTE-TINT)/(DXE+DXW)*(1.-HW0/HEPS)+TXW*HW0/HEPS
TXP=TXW
ELSEIF(T_FE) THEN
IF(I.NE.1 .AND. HE0.LT.HEPS)&
&       TXE=(TINT-TTW)/(DXE+DXW)*(1.-HE0/HEPS)+TXE*HE0/HEPS
TXP=TXE
ELSEIF(F(IM1,J,K).EQ.F(IP1,J,K)) THEN
TXP=0.5*(TXW+TXE)
ELSEIF(ABS(F(IM1,J,K)).LT.ABS(F(IP1,J,K))) THEN
TXP=TXW
ELSEIF(ABS(F(IM1,J,K)).GT.ABS(F(IP1,J,K))) THEN
TXP=TXE
ENDIF

IF(T_FS .AND. T_FN) THEN
TYP=0.5*(TYS+TYN)
ELSEIF(T_FS) THEN
IF(J.NE.NJ .AND. HS0.LT.HEPS)&
&       TYS=(TTN-TINT)/(DYN+DYS)*(1.-HS0/HEPS)+TYS*HS0/HEPS
TYP=TYS
ELSEIF(T_FN) THEN
IF(J.NE.1 .AND. HN0.LT.HEPS)&
&       TYN=(TINT-TTS)/(DYN+DYS)*(1.-HN0/HEPS)+TYN*HN0/HEPS
TYP=TYN
ELSEIF(F(I,JM1,K).EQ.F(I,JP1,K)) THEN
TYP=0.5*(TYS+TYN)
ELSEIF(ABS(F(I,JM1,K)).LT.ABS(F(I,JP1,K))) THEN
TYP=TYS
ELSEIF(ABS(F(I,JM1,K)).GT.ABS(F(I,JP1,K))) THEN
TYP=TYN
ENDIF

110     QN(I,J,K,L)=TXP*FNX(I,J,K)+TYP*FNY(I,J,K)
!-------------------------------------------------
IF(I23.EQ.2) GOTO 100

KKK=MAX(K,2)
DZB0=ZD(KKK)
DZT0=ZD(KP1)

DZB=DZB0
DZT=DZT0

IF(T_FB) HB0=F(I,J,K)/(F(I,J,K)-F(I,J,KM1))
IF(T_FT) HT0=F(I,J,K)/(F(I,J,K)-F(I,J,KP1))
IF(T_FB) DZB=DZB0*AMAX1(HEPS,HB0)
IF(T_FT) DZT=DZT0*AMAX1(HEPS,HT0)

TZP=0.

IF(T_FP) THEN
IF(T_IB) DZB1=DZB0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,J,KM1)))
```

```
IF(T_IT) DZT1=DZT0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,J,KP1)))
IF(T_IB) TZB=(TINT-TWAL)/AMAX1(HEPS*DZB0,DZB1-DZB)
IF(T_IT) TZT=(TWAL-TINT)/AMAX1(HEPS*DZT0,DZT1-DZT)
IF(T_IB) TZP=TZB
IF(T_IT) TZP=TZT
IF(T_IB .AND. T_IT) TZP=0.5*(TZB+TZT)
GOTO 120
ENDIF

IF(T_SB) DZB=DZB0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,J,KM1)))
IF(T_ST) DZT=DZT0*AMAX1(HEPS,S(I,J,K)/(S(I,J,K)-S(I,J,KP1)))

TTB=T(I,J,KM1)
TTT=T(I,J,KP1)
IF(T_FB) TTB=TINT
IF(T_FT) TTT=TINT

TZB=(TTP-TTB)/DZB
TZT=(TTT-TTP)/DZT

IF(T_FB .AND. T_FT) THEN
TZP=0.5*(TZB+TZT)
ELSEIF(T_FB) THEN
IF(K.NE.NK .AND. HB0.LT.HEPS)&
&       TZB=(TTT-TINT)/(DZT+DZB)*(1.-HB0/HEPS)+TZB*HB0/HEPS
TZP=TZB
ELSEIF(T_FT) THEN
IF(K.NE.1 .AND. HT0.LT.HEPS)&
&       TZT=(TINT-TTB)/(DZT+DZB)*(1.-HT0/HEPS)+TZT*HT0/HEPS
TZP=TZT
ELSEIF(F(I,J,KM1).EQ.F(I,J,KP1)) THEN
TZP=0.5*(TZB+TZT)
ELSEIF(ABS(F(I,J,KM1)).LT.ABS(F(I,J,KP1))) THEN
TZP=TZB
ELSEIF(ABS(F(I,J,KM1)).GT.ABS(F(I,J,KP1))) THEN
TZP=TZT
ENDIF

120     QN(I,J,K,L)=QN(I,J,K,L)+TZP*FNZ(I,J,K)
100     CONTINUE

CALL QN_EXT(L)
1000    CONTINUE

RETURN
END

!***********************************************************************
SUBROUTINE QN_SUB(I,J,K,IPHASE)
!***********************************************************************
USE PARAMETERS
USE IGRID
USE GRID1
```

```
USE GRIDB
USE LEVFN
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
USE ICHEK
USE CALC1
USE IJKMK
USE MPIVAR
IMPLICIT NONE
INTEGER, INTENT(IN) :: I,J,K,IPHASE
REAL (KIND=8) :: AX0,AC0,AY0,AZ0,FFX,FFY,FFZ
LOGICAL T_2PH,T_1PH
!***********************************************************************
!
IF(IMK(I,J,K).EQ.1) GOTO 100
FFX=F(I,J,K)*FNX(I,J,K)
IF((FFX.GE.0. .AND. I.EQ. 1) .OR.&
&     (FFX.LE.0. .AND. I.EQ.NI)) THEN
IMK(I,J,K)=1
ELSEIF(FFX.GE.0. .AND. MK0(I-1,J,K).LE.1) THEN
AX0=ABS(FNX(I,J,K))/XD(I)
AC(I,J,K)=AC(I,J,K)+AX0
SS(I,J,K)=SS(I,J,K)+AX0*QN(I-1,J,K,IPHASE)
IMK(I,J,K)=1
ELSEIF(FFX.LE.0. .AND. MK0(I+1,J,K).LE.1) THEN
AX0=ABS(FNX(I,J,K))/XD(I+1)
AC(I,J,K)=AC(I,J,K)+AX0
SS(I,J,K)=SS(I,J,K)+AX0*QN(I+1,J,K,IPHASE)
IMK(I,J,K)=1
ELSEIF(FFX.GE.0. .AND. F(I-1,J,K)*FNX(I-1,J,K).LE.0.) THEN
IMK(I,J,K)=1
ELSEIF(FFX.LE.0. .AND. F(I+1,J,K)*FNX(I+1,J,K).GE.0.) THEN
IMK(I,J,K)=1
ENDIF
!
100     IF(JMK(I,J,K).EQ.1) GOTO 200
FFY=F(I,J,K)*FNY(I,J,K)
IF((FFY.GE.0. .AND. J.EQ. 1) .OR.&
&     (FFY.LE.0. .AND. J.EQ.NJ)) THEN
JMK(I,J,K)=1
ELSEIF(FFY.GE.0. .AND. MK0(I,J-1,K).LE.1) THEN
AY0=ABS(FNY(I,J,K))/YD(J)
AC(I,J,K)=AC(I,J,K)+AY0
SS(I,J,K)=SS(I,J,K)+AY0*QN(I,J-1,K,IPHASE)
JMK(I,J,K)=1
ELSEIF(FFY.LE.0. .AND. MK0(I,J+1,K).LE.1) THEN
AY0=ABS(FNY(I,J,K))/YD(J+1)
AC(I,J,K)=AC(I,J,K)+AY0
SS(I,J,K)=SS(I,J,K)+AY0*QN(I,J+1,K,IPHASE)
JMK(I,J,K)=1
ELSEIF(FFY.GE.0. .AND. F(I,J-1,K)*FNY(I,J-1,K).LE.0.) THEN
JMK(I,J,K)=1
```

```
ELSEIF(FFY.LE.0. .AND. F(I,J+1,K)*FNY(I,J+1,K).GE.0.) THEN
JMK(I,J,K)=1
ENDIF

200     IF(I23.EQ.2 .OR. KMK(I,J,K).EQ.1) GOTO 300
FFZ=F(I,J,K)*FNZ(I,J,K)
IF((FFZ.GE.0. .AND. K.EQ. 1) .OR.&
&     (FFZ.LE.0. .AND. K.EQ.NK)) THEN
KMK(I,J,K)=1
ELSEIF(FFZ.GE.0. .AND. MK0(I,J,K-1).LE.1) THEN
AZ0=ABS(FNZ(I,J,K))/ZD(K)
AC(I,J,K)=AC(I,J,K)+AZ0
SS(I,J,K)=SS(I,J,K)+AZ0*QN(I,J,K-1,IPHASE)
KMK(I,J,K)=1
ELSEIF(FFZ.LE.0. .AND. MK0(I,J,K+1).LE.1) THEN
AZ0=ABS(FNZ(I,J,K))/ZD(K+1)
AC(I,J,K)=AC(I,J,K)+AZ0
SS(I,J,K)=SS(I,J,K)+AZ0*QN(I,J,K+1,IPHASE)
KMK(I,J,K)=1
ELSEIF(FFZ.GE.0. .AND. F(I,J,K-1)*FNZ(I,J,K-1).LE.0.) THEN
KMK(I,J,K)=1
ELSEIF(FFZ.LE.0. .AND. F(I,J,K+1)*FNZ(I,J,K+1).GE.0.) THEN
KMK(I,J,K)=1
ENDIF
!
300     IF(IMK(I,J,K).EQ.0 .OR. JMK(I,J,K).EQ.0) RETURN
IF(I23.EQ.3 .AND. KMK(I,J,K).EQ.0) RETURN
AC0=AMAX1(AC(I,J,K),1.E-30)
QN(I,J,K,IPHASE)=SS(I,J,K)/AC0
MK0(I,J,K)=1
!
RETURN
END

!**********************************************************************
SUBROUTINE RK_GET
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE IGRID
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE APANS
USE AEANS
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL (KIND=8) :: FYN,FYS,FBT,FZB,FZT,FXW,FXE,CNXE,CNXW,CNYN,CNYS,FX1,FY1,DET
!**********************************************************************
DO 100 K=K11,KNK
```

```
DO 100 J=1,NJ
DO 100 I=1,NI
IF(I .NE. 1) AW(I,J,K)=(F(I,J,K)-F(I-1,J,K))/XD(I)
IF(J .NE. 1) AS(I,J,K)=(F(I,J,K)-F(I,J-1,K))/YD(J)
IF(I23.EQ.2) GOTO 100
IF(K .NE. 1) AB(I,J,K)=(F(I,J,K)-F(I,J,K-1))/ZD(K)
100     CONTINUE

DO 110 K=2,NKM
DO 110 J=2,NJM
DO 110 I=2,NI
FYS=AS(I,J   ,K)*XDW(I)+AS(I-1,J   ,K)*XDE(I)
FYN=AS(I,J+1,K)*XDW(I)+AS(I-1,J+1,K)*XDE(I)
FY1=0.25*(FYS+FYN)**2
IF(I23.EQ.2) GOTO 115
FZB=AB(I,J,K   )*XDW(I)+AB(I-1,J,K   )*XDE(I)
FZT=AB(I,J,K+1)*XDW(I)+AB(I-1,J,K+1)*XDE(I)
FZ1=0.25*(FZB+FZT)**2
115     DET=AMAX1(SQRT(AW(I,J,K)**2+FY1+FZ1),1.E-10)
AE(I,J,K)=AW(I,J,K)/DET
110     CONTINUE

DO 120 K=2,NKM
DO 120 J=2,NJ
DO 120 I=2,NIM
FXW=AW(I   ,J,K)*YDS(J)+AW(I   ,J-1,K)*YDN(J)
FXE=AW(I+1,J,K)*YDS(J)+AW(I+1,J-1,K)*YDN(J)
FX1=0.25*(FXW+FXE)**2
IF(I23.EQ.2) GOTO 125
FZB=AB(I,J,K   )*YDS(J)+AB(I,J-1,K   )*YDN(J)
FZT=AB(I,J,K+1)*YDS(J)+AB(I,J-1,K+1)*YDN(J)
FZ1=0.25*(FZB+FZT)**2
125     DET=AMAX1(SQRT(FX1+AS(I,J,K)**2+FZ1),1.E-10)
AN(I,J,K)=AS(I,J,K)/DET
120     CONTINUE

IF(I23.EQ.2) GOTO 135
DO 130 K=2,NK
DO 130 J=2,NJM
DO 130 I=2,NIM
FXW=AW(I   ,J,K)*ZDB(K)+AW(I   ,J,K-1)*ZDT(K)
FXE=AW(I+1,J,K)*ZDB(K)+AW(I+1,J,K-1)*ZDT(K)
FX1=0.25*(FXW+FXE)**2
FYS=AS(I,J   ,K)*ZDB(K)+AS(I,J   ,K-1)*ZDT(K)
FYN=AS(I,J+1,K)*ZDB(K)+AS(I,J+1,K-1)*ZDT(K)
FY1=0.25*(FYS+FYN)**2
DET=AMAX1(SQRT(FX1+FY1+AB(I,J,K)**2),1.E-10)
AT(I,J,K)=AB(I,J,K)/DET
130     CONTINUE

135     DO 150 K=2,NKM
DO 150 J=2,NJM
DO 150 I=2,NIM
```

```
CNXE=RU(I+1)*AE(I+1,J,K)
CNXW=RU(I   )*AE(I   ,J,K)
CNYN=RP(I   )*AN(I,J+1,K)
CNYS=RP(I   )*AN(I,J   ,K)
RK(I,J,K)=((CNXE-CNXW)/XC(I)+(CNYN-CNYS)/YC(J))/RP(I)
IF(I23.EQ.2) GOTO 150
RK(I,J,K)=RK(I,J,K)+(AT(I,J,K+1)-AT(I,J,K))/ZC(K)
150     CONTINUE

RETURN
END

!***********************************************************************
SUBROUTINE SOLVE
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDMO
USE INDDT
USE IGRID
USE GRID1
USE GRIDB
USE CHCFL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE RESLT
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL (KIND=8) :: CF1,UMAX,VMAX,WMAX
!***********************************************************************
CF=1.E10
DO 10 K=2,NKM
DO 10 J=2,NJM
DO 10 I=2,NIM
UMAX=AMAX1(ABS(U(I,J,K)),ABS(U(I,J,K)+AMX(I,J,K)/RHOF(FU(I,J,K))))
VMAX=AMAX1(ABS(V(I,J,K)-VBUB),ABS(V(I,J,K)-VBUB+AMY(I,J,K)/RHOF(FV(I,J,K))))
WMAX=AMAX1(ABS(W(I,J,K)),ABS(W(I,J,K)+AMZ(I,J,K)/RHOF(FW(I,J,K))))
CF1=AMAX1(UMAX/XC(I),VMAX/YC(J),WMAX/ZC(K))
CF1=1./(CF1+1.E-10)
IF(CF1 .GT. CF ) GOTO 10
CF=CF1
ICF=I
JCF=J
KCF=K
10      CONTINUE
```

```
!
DT=AMIN1(DT0,CF*CFO)
IF(DT .LT. 1.E-8) WRITE(I69,*) ' DT IS TOO SMALL'
IF(DT .LT. 1.E-8) STOP
BDEFF=DEFF

DO 110 K=K11,KNK
DO 110 J=1,NJ
DO 110 I=1,NI
110     BF(I,J,K)=F(I,J,K)

DO 120 K=2,NKM
DO 120 J=2,NJM
DO 120 I=2,NIM
120     T(I,J,K)=AMAX1(0.,AMIN1(1.,T(I,J,K)))
CALL T_BC

CALL UL_GET
CALL TL_GET

IF(NBUB(1) .GE. 1) THEN
CALL IPBUB_GET
CALL BVOLB_MOD
CALL F_GET
CALL F_MOD
IF(JMOVE.EQ.1) CALL F_JMOVE
CALL IBUB_MOD
CALL H_GET
CALL N_GET
CALL RK_GET
ENDIF

CALL T_SOLV
CALL AM_GET

CALL U_SOLV
CALL V_SOLV
CALL W_SOLV

CALL DP_ADD
CALL P_SOLV
CALL PROJT

CALL U_BC
CALL V_BC
CALL W_BC

RETURN
END

!**********************************************************************
SUBROUTINE T_GEAS
!**********************************************************************
```

```
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L,NT
REAL (KIND=8) :: DXN,DXS,DXW,DXE,DZB,DZT,DYN,DYS
REAL (KIND=8) ::
HB0,HE0,HN0,HS0,HT0,HW0,TX,TY,TXP,TYP,TTB,TTE,TTN,TTP,TTS,TTW,TXE,TXW,TYN,TYS,TTT,T
ZB,TZP,TZT
REAL (KIND=8) ::
TTNN,TTSS,TTEE,TTBB,TYYN,TYYS,TYYP,TYNN,TYSS,TXXP,TXXW,TXXE,TXWW,TXEE,TTWW,TTT
T,TZZP,TZZB,TZTT,TZBB,TZZT
REAL (KIND=8) :: DZTT,DZC,DZBB,DYSS,DYNN,DYC,DXWW,DXC,DXEE
REAL (KIND=8) :: TKM,HZT,HZB,HXE,HXW,HX,HZ,HY,FLUX,FLX,FLY,FLZ=0.,TZ=0.
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT,T_2D,T_3D
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
LOGICAL T_FWW,T_FEE,T_FSS,T_FNN,T_FBB,T_FTT
LOGICAL T_SWW,T_SEE,T_SSS,T_SNN,T_SBB,T_STT
!**************************************************************************
!
T_2D=I23.EQ.2
T_3D=.NOT. T_2D

DO 10 K=KKST,KKEND
DO 10 J=JJST,JJEND
DO 10 I=IIST,IIEND
10      TSOR(I,J,K)=0.
!---------------------------------------------------------------------
DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND
DO 100 I=IIST,IIEND
IF(S(I,J,K).LE.0. .OR. F(I,J,K).EQ.0.) GOTO 100
IF(   IQ_VAP.EQ.0 .AND. F(I,J,K).LT.0.) GOTO 100
IF(   IQ_LIQ.EQ.0 .AND. F(I,J,K).GT.0.) GOTO 100
L=1
IF(F(I,J,K).GT.0.) L=2
!---------------------------------------------------------------------
```

```
FLX=0.5*(RU(I)*UL(I,J,K,L)+RU(I+1)*UL(I+1,J,K,L))
FLY=0.5*RP(I)*(VL(I,J,K,L)+VL(I,J+1,K,L))-RP(I)*VBUB
IF(T_3D) FLZ=0.5*(WL(I,J,K,L)+WL(I,J,K+1,L))
!---------------------------------------------------------------------
DXW=XD(I    )
DXE=XD(I+1)
DYS=YD(J    )
DYN=YD(J+1)

TTP=TL(I    ,J,K,L)
TTW=TL(I-1,J,K,L)
TTE=TL(I+1,J,K,L)
TTS=TL(I,J-1,K,L)
TTN=TL(I,J+1,K,L)

CALL DX_TT(I,I-1,J,K,DXW,TTW,T_FW,T_SW)
CALL DX_TT(I,I+1,J,K,DXE,TTE,T_FE,T_SE)
CALL DY_TT(I,J,J-1,K,DYS,TTS,T_FS,T_SS)
CALL DY_TT(I,J,J+1,K,DYN,TTN,T_FN,T_SN)

DXC=0.5*(DXW+DXE)
DYC=0.5*(DYS+DYN)

TKM=1./TKP(I,J,K)
AW(I,J,K)=TKM*RU(I    )/DXW/DXC
AE(I,J,K)=TKM*RU(I+1)/DXE/DXC
AS(I,J,K)=TKM*RP(I    )/DYS/DYC
AN(I,J,K)=TKM*RP(I    )/DYN/DYC
AB(I,J,K)=0.
AT(I,J,K)=0.
!---------------------------------------------------------------------
IF(T_2D) GOTO 110

DZB=ZD(K    )
DZT=ZD(K+1)
TTB=TL(I,J,K-1,L)
TTT=TL(I,J,K+1,L)

CALL DZ_TT(I,J,K,K-1,DZB,TTB,T_FB,T_SB)
CALL DZ_TT(I,J,K,K+1,DZT,TTT,T_FT,T_ST)

DZC=0.5*(DZB+DZT)
AB(I,J,K)=TKM/DZB/DZC
AT(I,J,K)=TKM/DZT/DZC
!---------------------------------------------------------------------
110     AP(I,J,K)=AE(I,J,K)+AW(I,J,K)+AN(I,J,K)+AS(I,J,K)+AB(I,J,K)+AT(I,J,K)
!---------------------------------------------------------------------
IF(T_FW .OR. T_SW) TSOR(I,J,K)=TSOR(I,J,K)+AW(I,J,K)*TTW
IF(T_FW .OR. T_SW) AW(I,J,K)=0.

IF(T_FE .OR. T_SE) TSOR(I,J,K)=TSOR(I,J,K)+AE(I,J,K)*TTE
IF(T_FE .OR. T_SE) AE(I,J,K)=0.
```

```
IF(T_FS .OR. T_SS) TSOR(I,J,K)=TSOR(I,J,K)+AS(I,J,K)*TTS
IF(T_FS .OR. T_SS) AS(I,J,K)=0.

IF(T_FN .OR. T_SN) TSOR(I,J,K)=TSOR(I,J,K)+AN(I,J,K)*TTN
IF(T_FN .OR. T_SN) AN(I,J,K)=0.

IF(T_2D) GOTO 120
IF(T_FB .OR. T_SB) TSOR(I,J,K)=TSOR(I,J,K)+AB(I,J,K)*TTB
IF(T_FB .OR. T_SB) AB(I,J,K)=0.

IF(T_FT .OR. T_ST) TSOR(I,J,K)=TSOR(I,J,K)+AT(I,J,K)*TTT
IF(T_FT .OR. T_ST) AT(I,J,K)=0.
120     CONTINUE
!------------------------------------------------- Convection
TXW=(TTP-TTW)/DXW
TXE=(TTE-TTP)/DXE
TYS=(TTP-TTS)/DYS
TYN=(TTN-TTP)/DYN

IF(FLX.GE.0.) TX=TXW
IF(FLX.LT.0.) TX=TXE
IF(FLY.GE.0.) TY=TYS
IF(FLY.LT.0.) TY=TYN

TXXP=(TXE-TXW)/(DXE+DXW)
TYYP=(TYN-TYS)/(DYN+DYS)

IF(T_2D) GOTO 130
TZB=(TTP-TTB)/DZB
TZT=(TTT-TTP)/DZT

IF(FLZ.GE.0.) TZ=TZB
IF(FLZ.LT.0.) TZ=TZT

TZZP=(TZT-TZB)/(DZT+DZB)
130     CONTINUE
!-------------------------------------------------
IF(FLX.GE.0.) THEN
IF(I.EQ.2 .OR. T_FW .OR. T_SW) THEN
TX=TX+DXE*TXXP
ELSE
DXWW=XD(I-1)
TTWW=TL(I-2,J,K,L)
CALL DX_TT(I-1,I-2,J,K,DXWW,TTWW,T_FWW,T_SWW)
TXWW=(TTW-TTWW)/DXWW
TXXW=(TXW-TXWW)/(DXW+DXWW)
TX=TX+DXE*AMIN2(TXXP,IENOT*TXXW)
ENDIF
ELSE
IF(I.EQ.NIM .OR. T_FE .OR. T_SE) THEN
TX=TX-DXW*TXXP
ELSE
DXEE=XD(I+2)
```

```
TTEE=TL(I+2,J,K,L)
CALL DX_TT(I+1,I+2,J,K,DXEE,TTEE,T_FEE,T_SEE)
TXEE=(TTEE-TTE)/DXEE
TXXE=(TXEE-TXE)/(DXEE+DXE)
TX=TX-DXW*AMIN2(TXXP,IENOT*TXXE)
ENDIF
ENDIF
!---------------------------------------------------------
IF(FLY.GE.0.) THEN
IF(J.EQ.2 .OR. T_FS .OR. T_SS) THEN
TY=TY+DYN*TYYP
ELSE
DYSS=YD(J-1)
TTSS=TL(I,J-2,K,L)
CALL DY_TT(I,J-1,J-2,K,DYSS,TTSS,T_FSS,T_SSS)
TYSS=(TTS-TTSS)/DYSS
TYYS=(TYS-TYSS)/(DYS+DYSS)
TY=TY+DYN*AMIN2(TYYP,IENOT*TYYS)
ENDIF
ELSE
IF(J.EQ.NJM .OR. T_FN .OR. T_SN) THEN
TY=TY-DYS*TYYP
ELSE
DYNN=YD(J+2)
TTNN=TL(I,J+2,K,L)
CALL DY_TT(I,J+1,J+2,K,DYNN,TTNN,T_FNN,T_SNN)
TYNN=(TTNN-TTN)/DYNN
TYYN=(TYNN-TYN)/(DYNN+DYN)
TY=TY-DYS*AMIN2(TYYP,IENOT*TYYN)
ENDIF
ENDIF

IF(T_2D) GOTO 150
!-------------------------------------------------
IF(FLZ.GE.0.) THEN
IF(K.EQ.2 .OR. T_FB .OR. T_SB) THEN
TZ=TZ+DZT*TZZP
ELSE
DZBB=ZD(K-1)
TTBB=TL(I,J,K-2,L)
CALL DZ_TT(I,J,K-1,K-2,DZBB,TTBB,T_FBB,T_SBB)
TZBB=(TTB-TTBB)/DZBB
TZZB=(TZB-TZBB)/(DZB+DZBB)
TZ=TZ+DZT*AMIN2(TZZP,IENOT*TZZB)
ENDIF
ELSE
IF(K.EQ.NKM .OR. T_FT .OR. T_ST) THEN
TZ=TZ-DZB*TZZP
ELSE
DZTT=ZD(K+2)
TTTT=TL(I,J,K+2,L)
CALL DZ_TT(I,J,K+1,K+2,DZTT,TTTT,T_FTT,T_STT)
TZTT=(TTTT-TTT)/DZTT
```

```
TZZT=(TZTT-TZT)/(DZTT+DZT)
TZ=TZ-DZB*AMIN2(TZZP,IENOT*TZZT)
ENDIF
ENDIF
!-----------------------------------------------------
150     FLUX=FLX*TX+FLY*TY+FLZ*TZ
TSOR(I,J,K)=TSOR(I,J,K)+RCP(I,J,K)*(RP(I)*TL(I,J,K,L)/DT-FLUX)
100     CONTINUE
!-----------------------------------------------------------------BC
IF(IBCW.NE.IWAL) THEN
DO 210 K=KKST,KKEND
DO 210 J=JJST,JJEND
AP(2,J,K)=AP(2,J,K)-AW(2,J,K)
210     AW(2,J,K)=0.
ENDIF
!
IF(IBCE.NE.IWAL) THEN
DO 220 K=KKST,KKEND
DO 220 J=JJST,JJEND
AP(NIM,J,K)=AP(NIM,J,K)-AE(NIM,J,K)
220     AE(NIM,J,K)=0.
ENDIF
!
IF(IBCS.NE.IWAL) THEN
DO 230 K=KKST,KKEND
DO 230 I=IIST,IIEND
AP(I,2,K)=AP(I,2,K)-AS(I,2,K)
230     AS(I,2,K)=0.
ENDIF

IF(IBCN.NE.IWAL) THEN
DO 240 K=KKST,KKEND
DO 240 I=IIST,IIEND
AP(I,NJM,K)=AP(I,NJM,K)-AN(I,NJM,K)
240     AN(I,NJM,K)=0.
ENDIF

IF(T_2D) GOTO 290
IF(IBCB.NE.IWAL) THEN
DO 250 J=JJST,JJEND
DO 250 I=IIST,IIEND
AP(I,J,2)=AP(I,J,2)-AB(I,J,2)
250     AB(I,J,2)=0.
ENDIF

IF(IBCN.NE.IWAL) THEN
DO 260 J=JJST,JJEND
DO 260 I=IIST,IIEND
AP(I,J,NKM)=AP(I,J,NKM)-AT(I,J,NKM)
260     AT(I,J,NKM)=0.
ENDIF
!-------------------------------------------
290     DO 300 K=KKST,KKEND
```

```
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND
AP(I,J,K)=AP(I,J,K)+RCP(I,J,K)*RP(I)/DT
300     APC(I,J,K)=AP(I,J,K)
!-------------------------------------------BCC
IF(IBCS.EQ.IWAL) THEN
DO 430 K=KKST,KKEND
DO 430 I=IIST,IIEND
HY=1.
IF(F(I,1,K).LE.0. .AND. IQ_VAP.EQ.0) HY=0.
IF(F(I,1,K).GE.0. .AND. IQ_LIQ.EQ.0) HY=0.
IF(T_2PH(F(I,1,K),F(I,2,K))) HY=F(I,1,K)/(F(I,1,K)-F(I,2,K))
HXW=0.5
HXE=0.5
IF(T_2PH(F(I,1,K),F(I-1,1,K))) HXW=XD(I    )*F(I,1,K)/(F(I,1,K)-F(I-1,1,K))/XC(I)
IF(T_2PH(F(I,1,K),F(I+1,1,K))) HXE=XD(I+1)*F(I,1,K)/(F(I,1,K)-F(I+1,1,K))/XC(I)
HX=HXW+HXE
HZ=1.
IF(I23.EQ.3) THEN
HZB=0.5
HZT=0.5
IF(T_2PH(F(I,1,K),F(I,1,K-1))) HZB=ZD(K    )*F(I,1,K)/(F(I,1,K)-F(I,1,K-1))/ZC(K)
IF(T_2PH(F(I,1,K),F(I,1,K+1))) HZT=ZD(K+1)*F(I,1,K)/(F(I,1,K)-F(I,1,K+1))/ZC(K)
HZ=HZB+HZT
ENDIF
AP(I, 2,K)=AP(I, 2,K)-AS(I, 2,K)*(1.-HY*HX*HZ)
430     CONTINUE
ENDIF

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(TSOR(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRE
CISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(TSOR(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(TSOR(1,1,1),SIZE(TSOR),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!***********************************************************************
SUBROUTINE T_GEAS_SUB
!***********************************************************************
```

```
USE PARAMETERS
USE PROPS
USE IGRID
USE GRID1
USE LEVFN
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K
REAL (KIND=8) :: TTW,TTS,TTB
REAL (KIND=8), INTENT(INOUT) :: DXW,DYS,DZB
INTEGER, INTENT(IN) :: I1,I2,J1,J2,K1,K2,IY,IZ,JX,JZ,KX,KY
LOGICAL T_FW,T_FS,T_FB,T_SW,T_SS,T_SB
!**********************************************************************
ENTRY DX_TT(I1,I2,JX,KX,DXW,TTW,T_FW,T_SW)
!**********************************************************************
J=JX
K=KX
T_FW=T_2PH(F(I1,J,K),F(I2,J,K))
T_SW=T_2PH(S(I1,J,K),S(I2,J,K))
IF(T_FW .AND. T_SW) THEN
T_FW=(F(I1,J,K)/(F(I1,J,K)-F(I2,J,K))).LT.&
&          (S(I1,J,K)/(S(I1,J,K)-S(I2,J,K)))
T_SW=.NOT. T_FW
ENDIF
IF(T_FW) THEN
DXW=DXW*AMAX1(HEPS,F(I1,J,K)/(F(I1,J,K)-F(I2,J,K)))
TTW=TINT
ENDIF
IF(T_SW) THEN
DXW=DXW*AMAX1(HEPS,S(I1,J,K)/(S(I1,J,K)-S(I2,J,K)))
TTW=TWAL
ENDIF
RETURN
!**********************************************************************
ENTRY DY_TT(IY,J1,J2,KY,DYS,TTS,T_FS,T_SS)
!**********************************************************************
I=IY
K=KY
T_FS=T_2PH(F(I,J1,K),F(I,J2,K))
T_SS=T_2PH(S(I,J1,K),S(I,J2,K))
IF(T_FS .AND. T_SS) THEN
T_FS=(F(I,J1,K)/(F(I,J1,K)-F(I,J2,K))).LT.&
&          (S(I,J1,K)/(S(I,J1,K)-S(I,J2,K)))
T_SS=.NOT. T_FS
ENDIF
IF(T_FS) THEN
DYS=DYS*AMAX1(HEPS,F(I,J1,K)/(F(I,J1,K)-F(I,J2,K)))
TTS=TINT
ENDIF
IF(T_SS) THEN
DYS=DYS*AMAX1(HEPS,S(I,J1,K)/(S(I,J1,K)-S(I,J2,K)))
TTS=TWAL
```

```
ENDIF
RETURN
!***********************************************************************
ENTRY DZ_TT(IZ,JZ,K1,K2,DZB,TTB,T_FB,T_SB)
!***********************************************************************
I=IZ
J=JZ
T_FB=T_2PH(F(I,J,K1),F(I,J,K2))
T_SB=T_2PH(S(I,J,K1),S(I,J,K2))
IF(T_FB .AND. T_SB) THEN
T_FB=(F(I,J,K1)/(F(I,J,K1)-F(I,J,K2))).LT.&
&          (S(I,J,K1)/(S(I,J,K1)-S(I,J,K2)))
T_SB=.NOT. T_FB
ENDIF
IF(T_FB) THEN
DZB=DZB*AMAX1(HEPS,F(I,J,K1)/(F(I,J,K1)-F(I,J,K2)))
TTB=TINT
ENDIF
IF(T_SB) THEN
DZB=DZB*AMAX1(HEPS,S(I,J,K1)/(S(I,J,K1)-S(I,J,K2)))
TTB=TWAL
ENDIF
RETURN
END

!***********************************************************************
SUBROUTINE TL_GET
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE IGRID
USE GRID1
USE LEVFN
USE ULVLL
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT
INTEGER :: ITMP,ITMPE,JTMP,JTMPE,KTMP,KTMPE
!***********************************************************************

DO 110 K=KKST,KKEND
DO 110 J=JJST,JJEND
DO 110 I=IIST,IIEND
TL(I,J,K,1)=T(I,J,K)
TL(I,J,K,2)=T(I,J,K)
110     CONTINUE

IF(IBOIL.EQ.1) THEN
DO 120 K=KKST,KKEND
DO 120 J=JJST,JJEND
DO 120 I=IIST,IIEND
```

```
IF(S(I,J,K).LE.0.) GOTO 120
IF(F(I,J,K).LE.0.) TL(I,J,K,2)=TINT
IF(F(I,J,K).GT.0.) TL(I,J,K,1)=TINT
120     CONTINUE
ENDIF

!----------------
RCCG(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)= TL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)
CALL GHNEIGHBOUR

ITMP=IIST-2
ITMPE=IIEND+2
JTMP=JJST-2
JTMPE=JJEND+2
KTMP=KKST-2
KTMPE=KKEND+2
IF (PIDX.EQ.0) ITMP=IIST
IF (PIDX.EQ.PROCSX-1) ITMPE=IIEND
IF (PIDY.EQ.0) JTMP=JJST
IF (PIDY.EQ.PROCSY-1) JTMPE=JJEND
IF (PIDZ.EQ.0) KTMP=KKST
IF (PIDZ.EQ.PROCSZ-1) KTMPE=KKEND
TL(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:2)=RCCG(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:
2)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(TL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2),SPANX*SPANY*SPANZ*2,MPI_DOUBLE_P
RECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(TL(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
),1:2),SIZECC(NT)*2,MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(TL(1,1,1,1),SIZE(TL),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
!--------------------


RETURN
END

!************************************************************************
SUBROUTINE U_GEAS
!************************************************************************
USE PARAMETERS
USE ICHOS
```

```
USE PROPS
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L,NT
INTEGER :: ITMP,ITMPE,JTMP,JTMPE,KTMP,KTMPE
REAL (KIND=8) ::
DXE,DXW,DYN,DYS,DZB,DZT,DIFB,DIFE,DIFN,DIFS,DIFT,DIFW,DUDX,DUIM,DVDY,DXEE,DXWW,DY
NN,DYSS,DZBB,DZTT
REAL (KIND=8) ::
RPU,UXE,UYN,UYS,UXW,UZB,UZT,UXEE,UXWW,UXXE,UXXP,UXXW,UYNN,UYSS,UYYN,UYYP,UYY
S,UZBB,UZTT,UZZB,UZZP,UZZT
REAL (KIND=8) :: FLUX,FLX,FLY,FLZ,UX,UY,UZ
REAL (KIND=8) :: VISB,VISE,VISM,VISN,VISS,VIST,VISW
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT,T_2D,T_3D
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
!***********************************************************************
T_2D=I23.EQ.2
T_3D=.NOT. T_2D

IF (PIDX.EQ.0) THEN
IIST = 3
ELSE IF (PIDX.EQ.PROCSX-1) THEN
IIEND = ID-1
ENDIF
IF (PIDY.EQ.0) THEN
JJST = 2
ELSE IF (PIDY.EQ.PROCSY-1) THEN
JJEND = JD-1
ENDIF
IF (PIDZ.EQ.0) THEN
KKST = 2
ELSE IF (PIDZ.EQ.PROCSZ-1) THEN
KKEND = KD-1
ENDIF

DO 10 K=KKST,KKEND
DO 10 J=JJST,JJEND
```

```
DO 10 I=IIST,IIEND
USOR(I,J,K)=0.
10       SS(I,J,K)=0.
!-------------------------------------------------------------------------
DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND
DO 100 I=IIST,IIEND
IF(SU(I,J,K).LE.0.) GOTO 100
L=1
IF(FU(I,J,K).GT.0.) L=2
!-------------------------------------------------------------------
FLX=0.25*( RU(I)*UL(I,J,K,L)+RU(I+1)*UL(I+1,J,K,L)+RU(I)*UL(I,J,K,L)+RU(I-1)*UL(I-1,J,K,L))
FLY=0.25*( RP(I    )*(VL(I    ,J+1,K,L)+VL(I,J,K,L))+RP(I-1)*(VL(I-1,J+1,K,L)+VL(I-1,J,K,L)))-RU(I)*VBUB
IF(T_3D)    FLZ=0.25*( WL(I,J,K+1,L)+WL(I,J,K,L)+WL(I-1,J,K+1,L)+WL(I-1,J,K,L))
!-------------------------------------------------------------------
VISW=1./VISMF(FU(I,J,K),FU(I-1,J,K),FV(I-1,J,K),FV(I-1,J+1,K),FW(I-1,J,K),FW(I-1,J,K+1))
VISE=1./VISMF(FU(I,J,K),FU(I+1,J,K),FV(I    ,J,K),FV(I    ,J+1,K),FW(I    ,J,K),FW(I    ,J,K+1))
VISS=1./VISMF(FU(I,J,K),FU(I,J-1,K),FV(I,J    ,K),FV(I-1,J    ,K),FC(I,J    ,K),FC(I,J    ,K+1))
VISN=1./VISMF(FU(I,J,K),FU(I,J+1,K),FV(I,J+1,K),FV(I-1,J+1,K),FC(I,J+1,K),FC(I,J+1,K+1))
IF(T_3D) THEN
VISB=1./VISMF(FU(I,J,K),FU(I,J,K-1),FW(I,J,K    ),FW(I-1,J,K    ),FC(I,J,K    ),FC(I,J+1,K    ))
VIST=1./VISMF(FU(I,J,K),FU(I,J,K+1),FW(I,J,K+1),FW(I-1,J,K+1),FC(I,J,K+1),FC(I,J+1,K+1))
ENDIF
!-------------------------------------------------------------------
DXW=XC(I-1)*FAC_EPS(SU(I,J,K),SU(I-1,J,K))
DXE=XC(I    )*FAC_EPS(SU(I,J,K),SU(I+1,J,K))
DYS=YD(J    )*FAC_EPS(SU(I,J,K),SU(I,J-1,K))
DYN=YD(J+1)*FAC_EPS(SU(I,J,K),SU(I,J+1,K))
IF(T_3D) THEN
DZB=ZD(K    )*FAC_EPS(SU(I,J,K),SU(I,J,K-1))
DZT=ZD(K+1)*FAC_EPS(SU(I,J,K),SU(I,J,K+1))
ENDIF
!---------------------------------------- Digonal Diffusion
AW(I,J,K)=VISW*RP(I-1)/DXW/XD(I)
AE(I,J,K)=VISE*RP(I    )/DXE/XD(I)
AS(I,J,K)=VISS*RU(I    )/DYS/YC(J)
AN(I,J,K)=VISN*RU(I    )/DYN/YC(J)
IF(T_3D) THEN
AB(I,J,K)=VISB/DZB/ZC(K)
AT(I,J,K)=VIST/DZT/ZC(K)
ENDIF
!---------------------------------------- OFF-Digonal Diffusion
!---------------------------------- [mr(du/dx)]e-w/dx
DIFW=VISW*RP(I-1)*(UL(I,J,K,L)-UL(I-1,J,K,L))/XC(I-1)
DIFE=VISE*RP(I    )*(UL(I+1,J,K,L)-UL(I,J,K,L))/XC(I    )
USOR(I,J,K)=USOR(I,J,K)+(DIFE-DIFW)/XD(I)
!---------------------------------- [mr(dv/dx)]n-s/dy
RPU=0.5*(RP(I)+RP(I-1))
DIFS=VISS*(VL(I,J,K,L)-VL(I-1,J,K,L))/XD(I)
DIFN=VISN*(VL(I,J+1,K,L)-VL(I-1,J+1,K,L))/XD(I)
USOR(I,J,K)=USOR(I,J,K)+RPU*(DIFN-DIFS)/YC(J)
!---------------------------------- [m(du/dx)+m(dv/dy)]
IF(IR.EQ.1) THEN
```

```
DUDX=0.5*( VISW*(UL(I,J,K,L)-UL(I-1,J,K,L))+VISE*(UL(I+1,J,K,L)-UL(I,J,K,L)))/XD(I)
DVDY=0.5*( VISW*(VL(I-1,J+1,K,L)-VL(I-1,J,K,L))+VISE*(VL(I,J+1,K,L)-VL(I,J,K,L)))/YC(J)
USOR(I,J,K)=USOR(I,J,K)+DUDX+DVDY
ENDIF
!---------------------------------- [m(dw/dx)]t-b/dz
IF(T_3D) THEN
DIFB=VISB*(WL(I,J,K    ,L)-WL(I-1,J,K    ,L))/XD(I)
DIFT=VIST*(WL(I,J,K+1,L)-WL(I-1,J,K+1,L))/XD(I)
USOR(I,J,K)=USOR(I,J,K)+(DIFT-DIFB)/ZC(K)
ENDIF
!---------------------------------------------------- Convection
UXW=(UL(I,J,K,L)-UL(I-1,J,K,L))/DXW
UXE=(UL(I+1,J,K,L)-UL(I,J,K,L))/DXE
UYS=(UL(I,J,K,L)-UL(I,J-1,K,L))/DYS
UYN=(UL(I,J+1,K,L)-UL(I,J,K,L))/DYN

IF(FLX.GE.0.) UX=UXW
IF(FLX.LT.0.) UX=UXE
IF(FLY.GE.0.) UY=UYS
IF(FLY.LT.0.) UY=UYN

IF(T_3D) THEN
UZB=(UL(I,J,K,L)-UL(I,J,K-1,L))/DZB
UZT=(UL(I,J,K+1,L)-UL(I,J,K,L))/DZT
IF(FLZ.GE.0.) UZ=UZB
IF(FLZ.LT.0.) UZ=UZT
ENDIF
IF(IENO.EQ.0) GOTO 150
!-----------------------------------------------------------
UXXP=(UXE-UXW)/(DXE+DXW)
UYYP=(UYN-UYS)/(DYN+DYS)
IF(T_3D) UZZP=(UZT-UZB)/(DZT+DZB)
!-----------------------------------------------------------
IF(FLX.GE.0.) THEN
IF(I.EQ.3 .OR. T_2PH(SU(I,J,K),SU(I-1,J,K))) THEN
UX=UX+DXE*UXXP
ELSE
DXWW=XC(I-2)*FAC_EPS(SU(I-1,J,K),SU(I-2,J,K))
UXWW=(UL(I-1,J,K,L)-UL(I-2,J,K,L))/DXWW
UXXW=(UXW-UXWW)/(DXW+DXWW)
UX=UX+DXE*AMIN2(UXXP,UXXW)
ENDIF
ELSE
IF(I.EQ.NIM .OR. T_2PH(SU(I,J,K),SU(I+1,J,K))) THEN
UX=UX-DXW*UXXP
ELSE
DXEE=XC(I+1)*FAC_EPS(SU(I+1,J,K),SU(I+2,J,K))
UXEE=(UL(I+2,J,K,L)-UL(I+1,J,K,L))/DXEE
UXXE=(UXEE-UXE)/(DXEE+DXE)
UX=UX-DXW*AMIN2(UXXP,UXXE)
ENDIF
ENDIF
!-----------------------------------------------------------
```

```
IF(FLY.GE.0.) THEN
IF(J.EQ.2 .OR. T_2PH(SU(I,J,K),SU(I,J-1,K))) THEN
UY=UY+DYN*UYYP
ELSE
DYSS=YD(J-1)*FAC_EPS(SU(I,J-1,K),SU(I,J-2,K))
UYSS=(UL(I,J-1,K,L)-UL(I,J-2,K,L))/DYSS
UYYS=(UYS-UYSS)/(DYS+DYSS)
UY=UY+DYN*AMIN2(UYYP,UYYS)
ENDIF
ELSE
IF(J.EQ.NJM .OR. T_2PH(SU(I,J,K),SU(I,J+1,K))) THEN
UY=UY-DYS*UYYP
ELSE
DYNN=YD(J+2)*FAC_EPS(SU(I,J+1,K),SU(I,J+2,K))
UYNN=(UL(I,J+2,K,L)-UL(I,J+1,K,L))/DYNN
UYYN=(UYNN-UYN)/(DYNN+DYN)
UY=UY-DYS*AMIN2(UYYP,UYYN)
ENDIF
ENDIF
!-----------------------------------------------------
IF(T_2D) GOTO 150
IF(FLZ.GE.0.) THEN
IF(K.EQ.2 .OR. T_2PH(SU(I,J,K),SU(I,J,K-1))) THEN
UZ=UZ+DZT*UZZP
ELSE
DZBB=ZD(K-1)*FAC_EPS(SU(I,J,K-1),SU(I,J,K-2))
UZBB=(UL(I,J,K-1,L)-UL(I,J,K-2,L))/DZBB
UZZB=(UZB-UZBB)/(DZB+DZBB)
UZ=UZ+DZT*AMIN2(UZZP,UZZB)
ENDIF
ELSE
IF(K.EQ.NKM .OR. T_2PH(SU(I,J,K),SU(I,J,K+1))) THEN
UZ=UZ-DZB*UZZP
ELSE
DZTT=ZD(K+2)*FAC_EPS(SU(I,J,K+1),SU(I,J,K+2))
UZTT=(UL(I,J,K+2,L)-UL(I,J,K+1,L))/DZTT
UZZT=(UZTT-UZT)/(DZTT+DZT)
UZ=UZ-DZB*AMIN2(UZZP,UZZT)
ENDIF
ENDIF
!-----------------------------------------------------
150     FLUX=FLX*UX+FLY*UY+FLZ*UZ
USOR(I,J,K)=USOR(I,J,K)+RHOU(I,J,K)*(RU(I)*UL(I,J,K,L)/DT-FLUX)
100     CONTINUE

!----------------------------------BC
DO 210 K=KKST,KKEND
DO 210 I=IIST,IIEND
IF(IBCS.EQ.IOUT .OR. IBCS.EQ.ISYM) AS(I,    2,K)=0.
IF(IBCN.EQ.IOUT .OR. IBCN.EQ.ISYM) AN(I,NJM,K)=0.
210     CONTINUE

IF(T_2D) GOTO 225
```

```
DO 220 J=JJST,JJEND
DO 220 I=IIST,IIEND
IF(IBCB.EQ.IOUT .OR. IBCB.EQ.ISYM) AB(I,J,   2)=0.
IF(IBCT.EQ.IOUT .OR. IBCT.EQ.ISYM) AT(I,J,NKM)=0.
220     CONTINUE
!-------------------------------------------------------------
225     IF(IBOIL.EQ.0) GOTO 390
DO 300 K=KKST,KKEND
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND
IF(T_2PH(FU(I,J,K),FU(I-1,J,K))) THEN
DUIM=(FU(I,J,K)*AMX(I-1,J,K)-FU(I-1,J,K)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I-1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AW(I,J,K)*DUIM
ENDIF

IF(T_2PH(FU(I,J,K),FU(I+1,J,K))) THEN
DUIM=(FU(I,J,K)*AMX(I+1,J,K)-FU(I+1,J,K)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I+1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AE(I,J,K)*DUIM
ENDIF
!
IF(T_2PH(FU(I,J,K),FU(I,J-1,K))) THEN
DUIM=(FU(I,J,K)*AMX(I,J-1,K)-FU(I,J-1,K)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I,J-1,K))*VFG
SS(I,J,K)=SS(I,J,K)+AS(I,J,K)*DUIM
ENDIF

IF(T_2PH(FU(I,J,K),FU(I,J+1,K))) THEN
DUIM=(FU(I,J,K)*AMX(I,J+1,K)-FU(I,J+1,K)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I,J+1,K))*VFG
SS(I,J,K)=SS(I,J,K)+AN(I,J,K)*DUIM
ENDIF

IF(T_2D) GOTO 300
IF(T_2PH(FU(I,J,K),FU(I,J,K-1))) THEN
DUIM=(FU(I,J,K)*AMX(I,J,K-1)-FU(I,J,K-1)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I,J,K-1))*VFG
SS(I,J,K)=SS(I,J,K)+AB(I,J,K)*DUIM
ENDIF

IF(T_2PH(FU(I,J,K),FU(I,J,K+1))) THEN
DUIM=(FU(I,J,K)*AMX(I,J,K+1)-FU(I,J,K+1)*AMX(I,J,K))/ABS(FU(I,J,K)-FU(I,J,K+1))*VFG
SS(I,J,K)=SS(I,J,K)+AT(I,J,K)*DUIM
ENDIF
300     CONTINUE
!-----------------------------------------
390     CONTINUE
DO 400 K=KKST,KKEND
DO 400 J=JJST,JJEND
DO 400 I=IIST,IIEND
USOR(I,J,K)=USOR(I,J,K)+SS(I,J,K)
VISM=1./VISF(FU(I,J,K))
AP(I,J,K)=RHOU(I,J,K)*RU(I)/DT+IR*VISM/RU(I)+AE(I,J,K)+AW(I,J,K)+AN(I,J,K)+AS(I,J,K)+AT(I,J,K)+AB
(I,J,K)
400     APC(I,J,K)=AP(I,J,K)

IF (TASKID.NE.0) THEN
```

```
CALL
MPI_SEND(USOR(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(USOR(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(USOR(1,1,1),SIZE(USOR),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE UL_GET
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE IGRID
USE LEVFN
USE ULVLL
USE QMSOR
USE UVTPP
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,NT
INTEGER :: ITMP,JTMP,KTMP,ITMPE,JTMPE,KTMPE
!**********************************************************************
DO 110 K=KKST,KKEND
DO 110 J=JJST,JJEND
DO 110 I=IIST,IIEND
UL(I,J,K,1)=U(I,J,K)
VL(I,J,K,1)=V(I,J,K)
WL(I,J,K,1)=W(I,J,K)
UL(I,J,K,2)=U(I,J,K)
VL(I,J,K,2)=V(I,J,K)
WL(I,J,K,2)=W(I,J,K)
110     CONTINUE
!
IF(IBOIL.EQ.1) THEN
DO 120 K=KKST,KKEND
DO 120 J=JJST,JJEND
DO 120 I=IIST,IIEND
IF(FU(I,J,K).LE.0.) UL(I,J,K,2)=U(I,J,K)+AMX(I,J,K)*VFG
IF(FU(I,J,K).GT.0.) UL(I,J,K,1)=U(I,J,K)-AMX(I,J,K)*VFG
```

```
IF(FV(I,J,K).LE.0.) VL(I,J,K,2)=V(I,J,K)+AMY(I,J,K)*VFG
IF(FV(I,J,K).GT.0.) VL(I,J,K,1)=V(I,J,K)-AMY(I,J,K)*VFG
IF(I23.EQ.2) GOTO 120
IF(FW(I,J,K).LE.0.) WL(I,J,K,2)=W(I,J,K)+AMZ(I,J,K)*VFG
IF(FW(I,J,K).GT.0.) WL(I,J,K,1)=W(I,J,K)-AMZ(I,J,K)*VFG
120     CONTINUE
ENDIF
!----------------
RCCG(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)= UL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)
CALL GHNEIGHBOUR

ITMP=IIST-2
ITMPE=IIEND+2
JTMP=JJST-2
JTMPE=JJEND+2
KTMP=KKST-2
KTMPE=KKEND+2
IF (PIDX.EQ.0) ITMP=IIST
IF (PIDX.EQ.PROCSX-1) ITMPE=IIEND
IF (PIDY.EQ.0) JTMP=JJST
IF (PIDY.EQ.PROCSY-1) JTMPE=JJEND
IF (PIDZ.EQ.0) KTMP=KKST
IF (PIDZ.EQ.PROCSZ-1) KTMPE=KKEND
UL(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:2)=RCCG(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:
2)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(UL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2),SPANX*SPANY*SPANZ*2,MPI_DOUBLE_P
RECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(UL(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
),1:2),SIZECC(NT)*2,MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(UL(1,1,1,1),SIZE(UL),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

!----------

RCCG(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)= VL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)
CALL GHNEIGHBOUR

ITMP=IIST-2
ITMPE=IIEND+2
JTMP=JJST-2
JTMPE=JJEND+2
```

```
KTMP=KKST-2
KTMPE=KKEND+2
IF (PIDX.EQ.0) ITMP=IIST
IF (PIDX.EQ.PROCSX-1) ITMPE=IIEND
IF (PIDY.EQ.0) JTMP=JJST
IF (PIDY.EQ.PROCSY-1) JTMPE=JJEND
IF (PIDZ.EQ.0) KTMP=KKST
IF (PIDZ.EQ.PROCSZ-1) KTMPE=KKEND
VL(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:2)=RCCG(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:
2)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(VL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2),SPANX*SPANY*SPANZ*2,MPI_DOUBLE_P
RECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(VL(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
),1:2),SIZECC(NT)*2,MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(VL(1,1,1,1),SIZE(VL),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

!-------------------
RCCG(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)= WL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2)
CALL GHNEIGHBOUR

ITMP=IIST-2
ITMPE=IIEND+2
JTMP=JJST-2
JTMPE=JJEND+2
KTMP=KKST-2
KTMPE=KKEND+2
IF (PIDX.EQ.0) ITMP=IIST
IF (PIDX.EQ.PROCSX-1) ITMPE=IIEND
IF (PIDY.EQ.0) JTMP=JJST
IF (PIDY.EQ.PROCSY-1) JTMPE=JJEND
IF (PIDZ.EQ.0) KTMP=KKST
IF (PIDZ.EQ.PROCSZ-1) KTMPE=KKEND
WL(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:2)=RCCG(ITMP:ITMPE,JTMP:JTMPE,KTMP:KTMPE,1:
2)

CALL RELOAD
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(WL(IIST:IIEND,JJST:JJEND,KKST:KKEND,1:2),SPANX*SPANY*SPANZ*2,MPI_DOUBLE_P
RECISION,0,TASKID,MPI_COMM_WORLD,IERR)
```

```
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(WL(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(N
T),1:2),SIZECC(NT)*2,MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(WL(1,1,1,1),SIZE(WL),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END

!**********************************************************************
SUBROUTINE UVPT_SOLVE
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE INDMO
USE INDDT
USE IGRID
USE MUGRD
USE GRID1
USE GRIDB
USE IDREF
USE SPERR
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE INDIT
USE RELXS
USE CHAR1
USE MISC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,N,ITER,MPII,MPJJ,MPKK
INTEGER :: ITMP,ITMPE,JTMP,JTMPE,KTMP,KTMPE,oldsize(3),newsize(3),starts(3),arr)
INTEGER :: NT,TCOUNT,DISPL(27)
!**************************************************************************************************************
ENTRY U_SOLV
!**********************************************************************
CALL CONTRACT(3,2,2,ID-1,JD-1,KD-1)
CALL RELOAD
!----------------------
```

```
N=1

CALL U_BC

CALL CONTRACT(3,2,2,ID-1,JD-1,KD-1)
CALL RELOAD

CALL U_GEAS
CALL U_BCAS

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=U(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
U(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)

CALL CONTRACT(3,2,2,ID-1,JD-1,KD-1)
CALL RELOAD

DO 110 K=KKST,KKEND
DO 110 J=JJST,JJEND
DO 110 I=IIST,IIEND
110
SS(I,J,K)=USOR(I,J,K)-APC(I,J,K)*U(I,J,K)+AE(I,J,K)*U(I+1,J,K)+AW(I,J,K)*U(I-1,J,K)+AN(I,J,K)*U(I,J+1,
K)+AS(I,J,K)*U(I,J-1,K)+AT(I,J,K)*U(I,J,K+1)+AB(I,J,K)*U(I,J,K-1)
CALL U_BCSS

IT(N)=-1
100     IT(N)=IT(N)+1
ITER=IT(N)

if (taskid.eq.0) then
IF(ITER    .EQ. ITER /JMONIT*JMONIT .AND.ITIME .EQ. ITIME/JMONT *JMONT) WRITE(I69, 9004)
CHT(N),ITER,U(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)
endif

IF( ITER .GE. 1 .AND. SSMAX(N) .LE. ESMAX.AND. FCMAX(N) .LE. EFMAX) THEN

if (taskid.eq.0) then
IF(ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,U(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)
endif

CALL U_BC

RETURN
ENDIF

IF(ITER .GE. MIT(N)) THEN

if (taskid.eq.0) then
WRITE(I69,*) ' ITERATION EXCEEDED FOR',CHT(N),ITIME
```

```
endif

STOP
ENDIF

CALL LNTDMA(N,ITER)
CALL UC_SET
CALL ITERM1(N,ITER)
CALL U_BC

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=U(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
U(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)

CALL CONTRACT(3,2,2,ID-1,JD-1,KD-1)
CALL RELOAD

DO 120 K=KKST,KKEND
DO 120 J=JJST,JJEND
DO 120 I=IIST,IIEND
120
SS(I,J,K)=USOR(I,J,K)-APC(I,J,K)*U(I,J,K)+AE(I,J,K)*U(I+1,J,K)+AW(I,J,K)*U(I-1,J,K)+AN(I,J,K)*U(I,J+1,
K)+AS(I,J,K)*U(I,J-1,K)+AT(I,J,K)*U(I,J,K+1)+AB(I,J,K)*U(I,J,K-1)

CALL U_BCSS
CALL ITERM2(N,ITER)

GOTO 100
!************************************************************************
ENTRY V_SOLV
!************************************************************************
CALL CONTRACT(2,3,2,ID-1,JD-1,KD-1)
CALL RELOAD
!--------------------
N=2

CALL V_BC
CALL V_GEAS
CALL V_BCAS

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=V(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
V(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)

CALL CONTRACT(2,3,2,ID-1,JD-1,KD-1)
CALL RELOAD
```

```
DO 210 K=KKST,KKEND
DO 210 J=JJST,JJEND
DO 210 I=IIST,IIEND
210
SS(I,J,K)=VSOR(I,J,K)-APC(I,J,K)*V(I,J,K)+AE(I,J,K)*V(I+1,J,K)+AW(I,J,K)*V(I-1,J,K)+AN(I,J,K)*V(I,J+1,
K)+AS(I,J,K)*V(I,J-1,K)+AT(I,J,K)*V(I,J,K+1)+AB(I,J,K)*V(I,J,K-1)
CALL V_BCSS

IT(N)=-1
200     IT(N)=IT(N)+1
ITER=IT(N)


IF(ITER.EQ.ITER/JMONIT*JMONIT.AND.ITIME.EQ.ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,V(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)

IF( ITER .GE. 1 .AND. SSMAX(N) .LE. ESMAX .AND. FCMAX(N) .LE. EFMAX) THEN
IF(ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,V(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)
CALL V_BC

RETURN
ENDIF

IF(ITER .GE. MIT(N)) THEN
WRITE(I69,*) ' ITERATION EXCEEDED FOR',CHT(N),ITIME
STOP
ENDIF

CALL LNTDMA(N,ITER)
CALL VC_SET
CALL ITERM1(N,ITER)
CALL V_BC

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=V(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
V(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)

CALL CONTRACT(2,3,2,ID-1,JD-1,KD-1)
CALL RELOAD

DO 220 K=KKST,KKEND
DO 220 J=JJST,JJEND
DO 220 I=IIST,IIEND
220     SS(I,J,K)=VSOR(I,J,K)-APC(I,J,K)*V(I,J,K)+AE(I,J,K)*V(I+1,J,K)+AW(I,J,K)*V(I-1,J,K)&
&               +AN(I,J,K)*V(I,J+1,K)+AS(I,J,K)*V(I,J-1,K)+AT(I,J,K)*V(I,J,K+1)+AB(I,J,K)*V(I,J,K-1)

CALL V_BCSS
CALL ITERM2(N,ITER)
```

```
GOTO 200
!************************************************************************
ENTRY W_SOLV
!************************************************************************
CALL CONTRACT(2,2,3,ID-1,JD-1,KD-1)
CALL RELOAD
!----------------------
IF(I23.EQ.2) RETURN
N=3

CALL W_BC
CALL W_GEAS
CALL W_BCAS
!
CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=W(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
W(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1
:KKEND+1)

CALL CONTRACT(2,2,3,ID-1,JD-1,KD-1)
CALL RELOAD

DO 310 K=KKST,KKEND
DO 310 J=JJST,JJEND
DO 310 I=IIST,IIEND
310
SS(I,J,K)=WSOR(I,J,K)-APC(I,J,K)*W(I,J,K)+AE(I,J,K)*W(I+1,J,K)+AW(I,J,K)*W(I-1,J,K)+AN(I,J,K)*W(I,J
+1,K)+AS(I,J,K)*W(I,J-1,K)+AT(I,J,K)*W(I,J,K+1)+AB(I,J,K)*W(I,J,K-1)
CALL W_BCSS

IT(N)=-1
300     IT(N)=IT(N)+1
ITER=IT(N)


IF(ITER    .EQ. ITER /JMONIT*JMONIT .AND.&
&     ITIME .EQ. ITIME/JMONT *JMONT)&
&WRITE(I69, 9004) CHT(N),ITER,W(IREF,JREF,KREF),SSMAX(N)&
&                                 ,FCMAX(N),RELAX(N),RELX2(N)
IF( ITER .GE. 1 .AND. SSMAX(N) .LE. ESMAX&
&                        .AND. FCMAX(N) .LE. EFMAX) THEN
IF(ITIME .EQ. ITIME/JMONT*JMONT)&
&WRITE(I69, 9004) CHT(N),ITER,W(IREF,JREF,KREF),SSMAX(N)&
&                                 ,FCMAX(N),RELAX(N),RELX2(N)
CALL W_BC

RETURN
ENDIF
IF(ITER .GE. MIT(N)) THEN
WRITE(I69,*) ' ITERATION EXCEEDED FOR',CHT(N),ITIME
STOP
```

```
ENDIF

CALL LNTDMA(N,ITER)
CALL WC_SET
CALL ITERM1(N,ITER)

CALL W_BC

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=W(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
W(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1
:KKEND+1)

CALL CONTRACT(2,2,3,ID-1,JD-1,KD-1)
CALL RELOAD

DO 320 K=KKST,KKEND
DO 320 J=JJST,JJEND
DO 320 I=IIST,IIEND
320
SS(I,J,K)=WSOR(I,J,K)-APC(I,J,K)*W(I,J,K)+AE(I,J,K)*W(I+1,J,K)+AW(I,J,K)*W(I-1,J,K)+AN(I,J,K)*W(I,J
+1,K)+AS(I,J,K)*W(I,J-1,K)+AT(I,J,K)*W(I,J,K+1)+AB(I,J,K)*W(I,J,K-1)

CALL W_BCSS
CALL ITERM2(N,ITER)

GOTO 300
!************************************************************************
ENTRY T_SOLV
!************************************************************************
CALL SHRINK
CALL RELOAD
!------------------------

IF(IBOIL.EQ.0) RETURN
N=4

CALL T_BC
CALL T_GEAS
CALL T_BCAS

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=T(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
T(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)
CALL SHRINK
CALL RELOAD

DO 410 K=KKST,KKEND
```

```
DO 410 J=JJST,JJEND
DO 410 I=IIST,IIEND
410
SS(I,J,K)=TSOR(I,J,K)-APC(I,J,K)*T(I,J,K)+AE(I,J,K)*T(I+1,J,K)+AW(I,J,K)*T(I-1,J,K)+AN(I,J,K)*T(I,J+1,K
)+AS(I,J,K)*T(I,J-1,K)+AT(I,J,K)*T(I,J,K+1)+AB(I,J,K)*T(I,J,K-1)

CALL T_BCSS
!--------------

CALL
MPI_TYPE_CREATE_SUBARRAY(3,OLDSIZE,NEWSIZE,STARTS,MPI_ORDER_FORTRAN,MPI_DOU
BLE_PRECISION,ARR,IERR)
CALL MPI_TYPE_COMMIT(ARR,IERR)

DO NT=0,26
DISPL(1)=0
IF (NT.GT.0) DISPL(NT+1)= DISPL(NT)+1
SIZECC(NT)=1
ENDDO

CALL
MPI_ALLGATHERV(rcc,SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECISION,SS(1,1,1),SIZECC,DISPL,
ARR,MPI_COMM_WORLD,IERR)
!--------------
IF (TASKID.NE.0) THEN
CALL
MPI_SEND(SS(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRECI
SION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(SS(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE(NT
)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL MPI_BCAST(SS(1,1,1),SIZE(SS),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

IT(N)=-1
400     IT(N)=IT(N)+1
ITER=IT(N)

if (taskid.eq.0) then
IF (ITER    .EQ. ITER /JMONIT*JMONIT .AND.ITIME .EQ. ITIME/JMONT *JMONT) WRITE(I69, 9004)
CHT(N),ITER,T(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)
endif

IF ( ITER .GE. 1 .AND. SSMAX(N) .LE. ESMAX.AND. FCMAX(N) .LE. EFMAX) THEN
if (taskid.eq.0) then
```

```
IF (ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,T(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)
endif

CALL T_BC

RETURN
ENDIF


IF(ITER .GE. MIT(N)) THEN
if (taskid.eq.0) then
WRITE(I69,*) ' ITERATION EXCEEDED FOR',CHT(N),ITIME
endif
STOP
ENDIF

CALL LNTDMA(N,ITER)
CALL TC_SET
CALL ITERM1(N,ITER)

CALL T_BC

CALL FRAGMENT
CALL RELOAD
RCC(IIST:IIEND,JJST:JJEND,KKST:KKEND)=T(IIST:IIEND,JJST:JJEND,KKST:KKEND)
CALL NEIGHBOUR
T(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:KKEND+1)=RCC(IIST-1:IIEND+1,JJST-1:JJEND+1,KKST-1:
KKEND+1)
CALL SHRINK
CALL RELOAD

DO 420 K= KKST,KKEND
DO 420 J= JJST,JJEND
DO 420 I= IIST,IIEND
420
SS(I,J,K)=TSOR(I,J,K)-APC(I,J,K)*T(I,J,K)+AE(I,J,K)*T(I+1,J,K)+AW(I,J,K)*T(I-1,J,K)+AN(I,J,K)*T(I,J+1,K
)+AS(I,J,K)*T(I,J-1,K)+AT(I,J,K)*T(I,J,K+1)+AB(I,J,K)*T(I,J,K-1)

CALL T_BCSS
CALL ITERM2(N,ITER)

GOTO 400

!************************************************************************
ENTRY P_SOLV
!************************************************************************
CALL SHRINK
CALL RELOAD

!--------------------
N=5
```

```
DO 510 K=K11,KNK
DO 510 J=1,NJ
DO 510 I=1,NI
SS(I,J,K)=0.
AW(I,J,K)=0.
AS(I,J,K)=0.
AB(I,J,K)=0.
AC(I,J,K)=0.
510     AP(I,J,K)=0.

CALL P_BC
CALL P_GEAS
CALL MU_GEAP

CALL M1PRTD(I23,MX(1),MY(1),MZ(1))
IF(MULT .GE. 2) CALL M2PRTD(I23,MX(2),MY(2),MZ(2))
IF(MULT .GE. 3) CALL M3PRTD(I23,MX(3),MY(3),MZ(3))
IF(MULT .GE. 4) CALL M4PRTD(I23,MX(4),MY(4),MZ(4))
IF(MULT .GE. 5) CALL M5PRTD(I23,MX(5),MY(5),MZ(5))
IF(MULT .GE. 6) CALL M6PRTD(I23,MX(6),MY(6),MZ(6))

DO 530 K=2,NKM
DO 530 J=2,NJM
DO 530 I=2,NIM
530
SS(I,J,K)=PSOR(I,J,K)-APC(I,J,K)*P(I,J,K)+AE(I,J,K)*P(I+1,J,K)+AW(I,J,K)*P(I-1,J,K)+AN(I,J,K)*P(I,J+1,
K)+AS(I,J,K)*P(I,J-1,K)+AT(I,J,K)*P(I,J,K+1)+AB(I,J,K)*P(I,J,K-1)

!-------------------------------------------------------
SUM=0.
DO 540 K=2,NKM
DO 540 J=2,NJM
DO 540 I=2,NIM
540     SUM=SUM+SS(I,J,K)
IF(ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69,9003) ' ***SUM=',SUM

IT(N)=-1
500     IT(N)=IT(N)+1

ITER=IT(N)


IF(ITER.EQ.ITER/JMONIT*JMONIT.AND.ITIME.EQ.ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,P(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)

IF( ITER.GE.1.AND.SSMAX(N).LE.ESMAX.AND. FCMAX(N) .LE. EFMAX) THEN

IF(ITIME .EQ. ITIME/JMONT*JMONT) WRITE(I69, 9004)
CHT(N),ITER,P(IREF,JREF,KREF),SSMAX(N),FCMAX(N),RELAX(N),RELX2(N)

CALL P_BC
RETURN
ENDIF
```

```
IF(ITER .GE. MIT(N)) THEN
WRITE(I69,*) ' ITERATION EXCEEDED FOR',CHT(N),ITIME
!         STOP
CALL P_BC
RETURN
ENDIF

CALL MU_GESS
DO 550 K=1,NK
DO 550 J=1,NJ
DO 550 I=1,NI
550     CC(I,J,K)=0.

IF(MULT .GE. 6) CALL M6VTDMA(I23,MX(6),MY(6),MZ(6))
IF(MULT .GE. 6) CALL MU_PRCC(I23,MX(6),MY(6),MZ(6),6)
IF(MULT .GE. 5) CALL M5VTDMA(I23,MX(5),MY(5),MZ(5))
IF(MULT .GE. 5) CALL MU_PRCC(I23,MX(5),MY(5),MZ(5),5)
IF(MULT .GE. 4) CALL M4VTDMA(I23,MX(4),MY(4),MZ(4))
IF(MULT .GE. 4) CALL MU_PRCC(I23,MX(4),MY(4),MZ(4),4)
IF(MULT .GE. 3) CALL M3VTDMA(I23,MX(3),MY(3),MZ(3))
IF(MULT .GE. 3) CALL MU_PRCC(I23,MX(3),MY(3),MZ(3),3)
IF(MULT .GE. 2) CALL M2VTDMA(I23,MX(2),MY(2),MZ(2))
IF(MULT .GE. 2) CALL MU_PRCC(I23,MX(2),MY(2),MZ(2),2)
CALL M1VTDMA(I23,MX(1),MY(1),MZ(1))

CALL SHRINK
CALL RELOAD

CALL ITERM1(N,ITER)

CALL P_BC
DO 560 K=2,NKM
DO 560 J=2,NJM
DO 560 I=2,NIM
560     SS(I,J,K)=PSOR(I,J,K)-APC(I,J,K)*P(I,J,K)+AE(I,J,K)*P(I+1,J,K)+AW(I,J,K)*P(I-1,J,K)&
&               +AN(I,J,K)*P(I,J+1,K)+AS(I,J,K)*P(I,J-1,K)+AT(I,J,K)*P(I,J,K+1)+AB(I,J,K)*P(I,J,K-1)

CALL ITERM2(N,ITER)
GOTO 500

9003    FORMAT(A,5(1PE12.4))
9004    FORMAT(A,I5,6(1PE12.4))
9005    FORMAT(I3,I3,6(1PE12.4))

END

!***********************************************************************
SUBROUTINE V_GEAS
!***********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
```

```
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L,NT
REAL (KIND=8) ::
DXE,DXW,DYN,DYS,DZB,DZT,DIFB,DIFE,DIFN,DIFS,DIFT,DIFW,DVIM,DVDY,DXEE,DXWW,DYNN,DY
SS,DZBB,DZTT
REAL (KIND=8) ::
RPU,VXE,VYN,VYS,VXW,VZB,VZT,VXEE,VXWW,VXXE,VXXP,VXXW,VYNN,VYSS,VYYN,VYYP,VYYS,
VZBB,VZTT,VZZB,VZZP,VZZT
REAL (KIND=8) :: FLUX,FLX,FLY,FLZ,VX,VY,VZ,FBDY,TMV
REAL (KIND=8) :: VISB,VISE,VISM,VISN,VISS,VIST,VISW
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT,T_2D,T_3D
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
!**************************************************************************
T_2D=I23.EQ.2
T_3D=.NOT. T_2D

DO 10 K=KKST,KKEND
DO 10 J=JJST,JJEND
DO 10 I=IIST,IIEND
TMV=T(I,J,K)*YDS(J)+T(I,J-1,K)*YDN(J)
FBDY=IGRAV*(-1.+BT0*DTWS*TMV)
IF(FV(I,J,K).LE.0.) FBDY=IGRAV*(-1.)
VSOR(I,J,K)=RP(I)*RHOV(I,J,K)*FBDY
10      SS(I,J,K)=0.
!-----------------------------------------------------------------------
DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND
DO 100 I=IIST,IIEND
IF(SV(I,J,K).LE.0.) GOTO 100
L=1
IF(FV(I,J,K).GT.0.) L=2
!-----------------------------------------------------------------------
FLX=0.25*( RU(I+1)*(UL(I+1,J,K,L)+UL(I+1,J-1,K,L))+RU(I   )*(UL(I   ,J,K,L)+UL(I   ,J-1,K,L)))
FLY=0.25*RP(I)*( VL(I,J+1,K,L)+2.*VL(I,J,K,L)+VL(I,J-1,K,L) )-RP(I)*VBUB
IF(T_3D) FLZ=0.25*( WL(I,J,K+1,L)+WL(I,J-1,K+1,L)+WL(I,J,K   ,L)+WL(I,J-1,K   ,L))
!-----------------------------------------------------------------------
```

```
VISW=1./VISMF(FV(I,J,K),FV(I-1,J,K),FU(I  ,J,K),FU(I  ,J-1,K),FC(I  ,J,K),FC(I  ,J,K+1))
VISE=1./VISMF(FV(I,J,K),FV(I+1,J,K),FU(I+1,J,K),FU(I+1,J-1,K),FC(I+1,J,K),FC(I+1,J,K+1))
VISS=1./VISMF(FV(I,J,K),FV(I,J-1,K),FU(I,J-1,K),FU(I+1,J-1,K),FW(I,J-1,K),FW(I,J-1,K+1))
VISN=1./VISMF(FV(I,J,K),FV(I,J+1,K),FU(I,J  ,K),FU(I+1,J  ,K),FW(I,J  ,K),FW(I,J  ,K+1))
IF(T_3D) THEN
VISB=1./VISMF(FV(I,J,K),FV(I,J,K-1),FW(I,J,K  ),FW(I,J-1,K  ),FC(I,J,K  ),FC(I+1,J,K  ))
VIST=1./VISMF(FV(I,J,K),FV(I,J,K+1),FW(I,J,K+1),FW(I,J-1,K+1),FC(I,J,K+1),FC(I+1,J,K+1))
ENDIF
!---------------------------------------------------------------------
DXW=XD(I  )*FAC_EPS(SV(I,J,K),SV(I-1,J,K))
DXE=XD(I+1)*FAC_EPS(SV(I,J,K),SV(I+1,J,K))
DYS=YC(J-1)*FAC_EPS(SV(I,J,K),SV(I,J-1,K))
DYN=YC(J  )*FAC_EPS(SV(I,J,K),SV(I,J+1,K))
IF(T_3D) THEN
DZB=ZD(K  )*FAC_EPS(SV(I,J,K),SV(I,J,K-1))
DZT=ZD(K+1)*FAC_EPS(SV(I,J,K),SV(I,J,K+1))
ENDIF
!---------------------------------------- Digonal Diffusion
AW(I,J,K)=VISW*RU(I  )/DXW/XC(I)
AE(I,J,K)=VISE*RU(I+1)/DXE/XC(I)
AS(I,J,K)=VISS*RP(I  )/DYS/YD(J)
AN(I,J,K)=VISN*RP(I  )/DYN/YD(J)
IF(T_3D) THEN
AB(I,J,K)=VISB/DZB/ZC(K)
AT(I,J,K)=VIST/DZT/ZC(K)
ENDIF
!---------------------------------------- Off-Digonal Diffusion
!------------------------------------- [mr(du/dy)]e-w/dx
DIFW=VISW*RU(I  )*(UL(I  ,J,K,L)-UL(I  ,J-1,K,L))/YD(J)
DIFE=VISE*RU(I+1)*(UL(I+1,J,K,L)-UL(I+1,J-1,K,L))/YD(J)
VSOR(I,J,K)=VSOR(I,J,K)+(DIFE-DIFW)/XC(I)
!------------------------------------- [mr(dv/dy)]n-s/dy
DIFS=VISS*RP(I)*(VL(I,J  ,K,L)-VL(I,J-1,K,L))/YC(J-1)
DIFN=VISN*RP(I)*(VL(I,J+1,K,L)-VL(I,J  ,K,L))/YC(J  )
VSOR(I,J,K)=VSOR(I,J,K)+(DIFN-DIFS)/YD(J)
!------------------------------------- [m(dw/dy)]t-b/dz
IF(T_3D) THEN
DIFB=VISB*(WL(I,J,K  ,L)-WL(I,J-1,K  ,L))/YD(J)
DIFT=VIST*(WL(I,J,K+1,L)-WL(I,J-1,K+1,L))/YD(J)
VSOR(I,J,K)=VSOR(I,J,K)+(DIFT-DIFB)/ZC(K)
ENDIF
!---------------------------------------------- Convection
VXW=(VL(I,J,K,L)-VL(I-1,J,K,L))/DXW
VXE=(VL(I+1,J,K,L)-VL(I,J,K,L))/DXE
VYS=(VL(I,J,K,L)-VL(I,J-1,K,L))/DYS
VYN=(VL(I,J+1,K,L)-VL(I,J,K,L))/DYN

IF(FLX.GE.0.) VX=VXW
IF(FLX.LT.0.) VX=VXE
IF(FLY.GE.0.) VY=VYS
IF(FLY.LT.0.) VY=VYN
!---------------------------------------
IF(T_3D) THEN
```

```
VZB=(VL(I,J,K,L)-VL(I,J,K-1,L))/DZB
VZT=(VL(I,J,K+1,L)-VL(I,J,K,L))/DZT
!
IF(FLZ.GE.0.) VZ=VZB
IF(FLZ.LT.0.) VZ=VZT
ENDIF
IF(IENO.EQ.0) GOTO 150
!---------------------------------------
VXXP=(VXE-VXW)/(DXE+DXW)
VYYP=(VYN-VYS)/(DYN+DYS)
IF(T_3D) VZZP=(VZT-VZB)/(DZT+DZB)
!---------------------------------- X DIRECTION
IF(FLX.GE.0.) THEN
IF(I.EQ.2 .OR. T_2PH(SV(I,J,K),SV(I-1,J,K))) THEN
VX=VX+DXE*VXXP
ELSE
DXWW=XD(I-1)*FAC_EPS(SV(I-1,J,K),SV(I-2,J,K))
VXWW=(VL(I-1,J,K,L)-VL(I-2,J,K,L))/DXWW
VXXW=(VXW-VXWW)/(DXW+DXWW)
VX=VX+DXE*AMIN2(VXXP,VXXW)
ENDIF
ELSE
IF(I.EQ.NIM .OR. T_2PH(SV(I,J,K),SV(I+1,J,K))) THEN
VX=VX-DXW*VXXP
ELSE
DXEE=XD(I+2)*FAC_EPS(SV(I+1,J,K),SV(I+2,J,K))
VXEE=(VL(I+2,J,K,L)-VL(I+1,J,K,L))/DXEE
VXXE=(VXEE-VXE)/(DXEE+DXE)
VX=VX-DXW*AMIN2(VXXP,VXXE)
ENDIF
ENDIF
!-------------------------------------------------------
IF(FLY.GE.0.) THEN
IF(J.EQ.3 .OR. T_2PH(SV(I,J,K),SV(I,J-1,K))) THEN
VY=VY+DYN*VYYP
ELSE
DYSS=YC(J-2)*FAC_EPS(SV(I,J-1,K),SV(I,J-2,K))
VYSS=(VL(I,J-1,K,L)-VL(I,J-2,K,L))/DYSS
VYYS=(VYS-VYSS)/(DYS+DYSS)
VY=VY+DYN*AMIN2(VYYP,VYYS)
ENDIF
ELSE
IF(J.EQ.NJM .OR. T_2PH(SV(I,J,K),SV(I,J+1,K))) THEN
VY=VY-DYS*VYYP
ELSE
DYNN=YC(J+1)*FAC_EPS(SV(I,J+1,K),SV(I,J+2,K))
VYNN=(VL(I,J+2,K,L)-VL(I,J+1,K,L))/DYNN
VYYN=(VYNN-VYN)/(DYNN+DYN)
VY=VY-DYS*AMIN2(VYYP,VYYN)
ENDIF
ENDIF
!-------------------------------------------------------
IF(T_2D) GOTO 150
```

```
IF(FLZ.GE.0.) THEN
IF(K.EQ.2 .OR. T_2PH(SV(I,J,K),SV(I,J,K-1))) THEN
VZ=VZ+DZT*VZZP
ELSE
DZBB=ZD(K-1)*FAC_EPS(SV(I,J,K-1),SV(I,J,K-2))
VZBB=(VL(I,J,K-1,L)-VL(I,J,K-2,L))/DZBB
VZZB=(VZB-VZBB)/(DZB+DZBB)
VZ=VZ+DZT*AMIN2(VZZP,VZZB)
ENDIF
ELSE
IF(K.EQ.NKM .OR. T_2PH(SV(I,J,K),SV(I,J,K+1))) THEN
VZ=VZ-DZB*VZZP
ELSE
DZTT=ZD(K+2)*FAC_EPS(SV(I,J,K+1),SV(I,J,K+2))
VZTT=(VL(I,J,K+2,L)-VL(I,J,K+1,L))/DZTT
VZZT=(VZTT-VZT)/(DZTT+DZT)
VZ=VZ-DZB*AMIN2(VZZP,VZZT)
ENDIF
ENDIF
!-----------------------------------------------------------
150     FLUX=FLX*VX+FLY*VY+FLZ*VZ
VSOR(I,J,K)=VSOR(I,J,K)+RHOV(I,J,K)*(RP(I)*VL(I,J,K,L)/DT-FLUX)
100     CONTINUE
!-------------------------------------------------------------BC
DO 210 K=KKST,KKEND
DO 210 J=JJST,JJEND
IF(IBCW.EQ.IOUT .OR. IBCW.EQ.ISYM) AW(    2,J,K)=0.
IF(IBCE.EQ.IOUT .OR. IBCE.EQ.ISYM) AE(NIM,J,K)=0.
210     CONTINUE

IF(T_2D) GOTO 225
DO 220 J=JJST,JJEND
DO 220 I=IIST,IIEND
IF(IBCB.EQ.IOUT .OR. IBCB.EQ.ISYM) AB(I,J,    2)=0.
IF(IBCT.EQ.IOUT .OR. IBCT.EQ.ISYM) AT(I,J,NKM)=0.
220     CONTINUE
!-----------------------------------------------------------
225     IF(IBOIL.EQ.0) GOTO 390
DO 300 K=KKST,KKEND
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND
IF(T_2PH(FV(I,J,K),FV(I-1,J,K))) THEN
DVIM=(FV(I,J,K)*AMY(I-1,J,K)-FV(I-1,J,K)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I-1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AW(I,J,K)*DVIM
ENDIF
!
IF(T_2PH(FV(I,J,K),FV(I+1,J,K))) THEN
DVIM=(FV(I,J,K)*AMY(I+1,J,K)-FV(I+1,J,K)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I+1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AE(I,J,K)*DVIM
ENDIF
```

```
IF(T_2PH(FV(I,J,K),FV(I,J-1,K))) THEN
DVIM=(FV(I,J,K)*AMY(I,J-1,K)-FV(I,J-1,K)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I,J-1,K))*VFG
SS(I,J,K)=SS(I,J,K)+AS(I,J,K)*DVIM
ENDIF

IF(T_2PH(FV(I,J,K),FV(I,J+1,K))) THEN
DVIM=(FV(I,J,K)*AMY(I,J+1,K)-FV(I,J+1,K)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I,J+1,K))*VFG
SS(I,J,K)=SS(I,J,K)+AN(I,J,K)*DVIM
ENDIF

IF(T_2D) GOTO 300
IF(T_2PH(FV(I,J,K),FV(I,J,K-1))) THEN
DVIM=(FV(I,J,K)*AMY(I,J,K-1)-FV(I,J,K-1)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I,J,K-1))*VFG
SS(I,J,K)=SS(I,J,K)+AB(I,J,K)*DVIM
ENDIF

IF(T_2PH(FV(I,J,K),FV(I,J,K+1))) THEN
DVIM=(FV(I,J,K)*AMY(I,J,K+1)-FV(I,J,K+1)*AMY(I,J,K))&
&       /ABS(FV(I,J,K)-FV(I,J,K+1))*VFG
SS(I,J,K)=SS(I,J,K)+AT(I,J,K)*DVIM
ENDIF
300     CONTINUE
!-------------------------------------------------------------
390     CONTINUE
DO 400 K=KKST,KKEND
DO 400 J=JJST,JJEND
DO 400 I=IIST,IIEND
VSOR(I,J,K)=VSOR(I,J,K)+SS(I,J,K)
AP(I,J,K)=RHOV(I,J,K)*RP(I)/DT+AE(I,J,K)+AW(I,J,K)+AN(I,J,K)+AS(I,J,K)+AT(I,J,K)+AB(I,J,K)
400     APC(I,J,K)=AP(I,J,K)

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(VSOR(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PRE
CISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(VSOR(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZE
(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(VSOR(1,1,1),SIZE(VSOR),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)
RETURN
END
```

```
!**********************************************************************
SUBROUTINE W_GEAS
!**********************************************************************
USE PARAMETERS
USE ICHOS
USE PROPS
USE INDDT
USE IGRID
USE IBNDS
USE GRID1
USE GRIDB
USE GRIDL
USE LEVFN
USE ULVLL
USE FFSOR
USE ROGAM
USE QMSOR
USE FNXYZ
USE APANS
USE AEANS
USE UVTPP
USE CALC1
USE FUNC
USE MPIVAR
IMPLICIT NONE
INTEGER :: I,J,K,L,NT
REAL (KIND=8) ::
DXW,DXE,DYN,DYS,DZT,DZB,DYSS,DYNN,DZTT,DZBB,DXWW,DXEE,DWIM,DIFW,DIFT,DIFS,DIFN,
DIFE,DIFB
REAL (KIND=8) ::
WX,WY,WZ,WXW,WXE,WZT,WZB,WYN,WYS,WXXW,WXXP,WXXE,WXEE,WXWW,WYNN,WYYS,WY
YP,WYYN,WYSS,WZBB,WZZT,WZZP,WZZB,WZTT
REAL (KIND=8) :: VIST,VISB,VISE,VISW,VISN,VISS,FLX,FLZ,FLY,FLUX
LOGICAL T_FW,T_FE,T_FS,T_FN,T_FB,T_FT,T_2D,T_3D
LOGICAL T_SW,T_SE,T_SS,T_SN,T_SB,T_ST
!**********************************************************************
DO 10 K=KKST,KKEND
DO 10 J=JJST,JJEND
DO 10 I=IIST,IIEND
WSOR(I,J,K)=0.
10      SS(I,J,K)=0.
!----------------------------------------------------------------------
DO 100 K=KKST,KKEND
DO 100 J=JJST,JJEND
DO 100 I=IIST,IIEND
IF(SW(I,J,K).LE.0.) GOTO 100
L=1
IF(FW(I,J,K).GT.0.) L=2
!----------------------------------------------------------------------
FLX=0.25*( UL(I+1,J,K,L)+UL(I+1,J,K-1,L)&
```

```
&               +UL(I    ,J,K,L)+UL(I    ,J,K-1,L))
FLY=0.25*( VL(I,J+1,K,L)+VL(I,J+1,K-1,L)&
&               +VL(I,J    ,K,L)+VL(I,J    ,K-1,L))-VBUB
FLZ=0.25*( WL(I,J,K+1,L)+2.*WL(I,J,K,L)+WL(I,J,K-1,L) )
!---------------------------------------------------------------------
VISW=1./VISMF(FW(I,J,K),FW(I-1,J,K),FU(I    ,J,K),FU(I    ,J,K-1)&
&                                   ,FC(I    ,J,K),FC(I    ,J+1,K))
VISE=1./VISMF(FW(I,J,K),FW(I+1,J,K),FU(I+1,J,K),FU(I+1,J,K-1)&
&                                   ,FC(I+1,J,K),FC(I+1,J+1,K))
VISS=1./VISMF(FW(I,J,K),FW(I,J-1,K),FV(I,J    ,K),FV(I,J    ,K-1)&
&                                   ,FC(I,J    ,K),FC(I+1,J    ,K))
VISN=1./VISMF(FW(I,J,K),FW(I,J+1,K),FV(I,J+1,K),FV(I,J+1,K-1)&
&                                   ,FC(I,J+1,K),FC(I+1,J+1,K))
VISB=1./VISMF(FW(I,J,K),FW(I,J,K-1),FU(I,J,K-1),FU(I+1,J,K-1)&
&                                   ,FV(I,J,K-1),FV(I,J+1,K-1))
VIST=1./VISMF(FW(I,J,K),FW(I,J,K+1),FU(I,J,K    ),FU(I+1,J,K    )&
&                                   ,FV(I,J,K    ),FV(I,J+1,K    ))
!---------------------------------------------------------------------
DXW=XD(I    )*FAC_EPS(SW(I,J,K),SW(I-1,J,K))
DXE=XD(I+1)*FAC_EPS(SW(I,J,K),SW(I+1,J,K))
DYS=YD(J    )*FAC_EPS(SW(I,J,K),SW(I,J-1,K))
DYN=YD(J+1)*FAC_EPS(SW(I,J,K),SW(I,J+1,K))
DZB=ZC(K-1)*FAC_EPS(SW(I,J,K),SW(I,J,K-1))
DZT=ZC(K    )*FAC_EPS(SW(I,J,K),SW(I,J,K+1))
!--------------------------------------- Digonal Diffusion
AW(I,J,K)=VISW/DXW/XC(I)
AE(I,J,K)=VISE/DXE/XC(I)
AS(I,J,K)=VISS/DYS/YC(J)
AN(I,J,K)=VISN/DYN/YC(J)
AB(I,J,K)=VISB/DZB/ZD(K)
AT(I,J,K)=VIST/DZT/ZD(K)
!--------------------------------------- Off-Digonal Diffusion
!----------------------------------- [m(du/dz)]e-w/dx
DIFW=VISW*(UL(I    ,J,K,L)-UL(I    ,J,K-1,L))/ZD(K)
DIFE=VISE*(UL(I+1,J,K,L)-UL(I+1,J,K-1,L))/ZD(K)
WSOR(I,J,K)=WSOR(I,J,K)+(DIFE-DIFW)/XC(I)
!----------------------------------- [m(dv/dz)]n-s/dy
DIFS=VISS*(VL(I,J    ,K,L)-VL(I,J    ,K-1,L))/ZD(K)
DIFN=VISN*(VL(I,J+1,K,L)-VL(I,J+1,K-1,L))/ZD(K)
WSOR(I,J,K)=WSOR(I,J,K)+(DIFN-DIFS)/YC(J)
!----------------------------------- [m(dw/dz)]t-b/dz
DIFB=VISB*(WL(I,J,K    ,L)-WL(I,J,K-1,L))/ZC(K-1)
DIFT=VIST*(WL(I,J,K+1,L)-WL(I,J,K    ,L))/ZC(K)
WSOR(I,J,K)=WSOR(I,J,K)+(DIFT-DIFB)/ZD(K)
!---------------------------------------------- Convection
WXW=(WL(I,J,K,L)-WL(I-1,J,K,L))/DXW
WXE=(WL(I+1,J,K,L)-WL(I,J,K,L))/DXE
WYS=(WL(I,J,K,L)-WL(I,J-1,K,L))/DYS
WYN=(WL(I,J+1,K,L)-WL(I,J,K,L))/DYN
WZB=(WL(I,J,K,L)-WL(I,J,K-1,L))/DZB
WZT=(WL(I,J,K+1,L)-WL(I,J,K,L))/DZT

IF(FLX.GE.0.) WX=WXW
```

```
IF(FLX.LT.0.) WX=WXE
IF(FLY.GE.0.) WY=WYS
IF(FLY.LT.0.) WY=WYN
IF(FLZ.GE.0.) WZ=WZB
IF(FLZ.LT.0.) WZ=WZT
IF(IENO.EQ.0) GOTO 150
!-------------------------------------------------------------
WXXP=(WXE-WXW)/(DXE+DXW)
WYYP=(WYN-WYS)/(DYN+DYS)
WZZP=(WZT-WZB)/(DZT+DZB)
!---------------------------------- X DIRECTION
IF(FLX.GE.0.) THEN
IF(I.EQ.2 .OR. T_2PH(SW(I,J,K),SW(I-1,J,K))) THEN
WX=WX+DXE*WXXP
ELSE
DXWW=XD(I-1)*FAC_EPS(SW(I-1,J,K),SW(I-2,J,K))
WXWW=(WL(I-1,J,K,L)-WL(I-2,J,K,L))/DXWW
WXXW=(WXW-WXWW)/(DXW+DXWW)
WX=WX+DXE*AMIN2(WXXP,WXXW)
ENDIF
ELSE
IF(I.EQ.NIM .OR. T_2PH(SW(I,J,K),SW(I+1,J,K))) THEN
WX=WX-DXW*WXXP
ELSE
DXEE=XD(I+2)*FAC_EPS(SW(I+1,J,K),SW(I+2,J,K))
WXEE=(WL(I+2,J,K,L)-WL(I+1,J,K,L))/DXEE
WXXE=(WXEE-WXE)/(DXEE+DXE)
WX=WX-DXW*AMIN2(WXXP,WXXE)
ENDIF
ENDIF
!-------------------------------------------------------
IF(FLY.GE.0.) THEN
IF(J.EQ.2 .OR. T_2PH(SW(I,J,K),SW(I,J-1,K))) THEN
WY=WY+DYN*WYYP
ELSE
DYSS=YD(J-1)*FAC_EPS(SW(I,J-1,K),SW(I,J-2,K))
WYSS=(WL(I,J-1,K,L)-WL(I,J-2,K,L))/DYSS
WYYS=(WYS-WYSS)/(DYS+DYSS)
WY=WY+DYN*AMIN2(WYYP,WYYS)
ENDIF
ELSE
IF(J.EQ.NJM .OR. T_2PH(SW(I,J,K),SW(I,J+1,K))) THEN
WY=WY-DYS*WYYP
ELSE
DYNN=YD(J+2)*FAC_EPS(SW(I,J+1,K),SW(I,J+2,K))
WYNN=(WL(I,J+2,K,L)-WL(I,J+1,K,L))/DYNN
WYYN=(WYNN-WYN)/(DYNN+DYN)
WY=WY-DYS*AMIN2(WYYP,WYYN)
ENDIF
ENDIF
!-------------------------------------------------------
IF(FLZ.GE.0.) THEN
IF(K.EQ.3 .OR. T_2PH(SW(I,J,K),SW(I,J,K-1))) THEN
```

```
WZ=WZ+DZT*WZZP
ELSE
DZBB=ZC(K-2)*FAC_EPS(SW(I,J,K-1),SW(I,J,K-2))
WZBB=(WL(I,J,K-1,L)-WL(I,J,K-2,L))/DZBB
WZZB=(WZB-WZBB)/(DZB+DZBB)
WZ=WZ+DZT*AMIN2(WZZP,WZZB)
ENDIF
ELSE
IF(K.EQ.NKM .OR. T_2PH(SW(I,J,K),SW(I,J,K+1))) THEN
WZ=WZ-DZB*WZZP
ELSE
DZTT=ZC(K+1)*FAC_EPS(SW(I,J,K+1),SW(I,J,K+2))
WZTT=(WL(I,J,K+2,L)-WL(I,J,K+1,L))/DZTT
WZZT=(WZTT-WZT)/(DZTT+DZT)
WZ=WZ-DZB*AMIN2(WZZP,WZZT)
ENDIF
ENDIF
!-----------------------------------------------------------
150     FLUX=FLX*WX+FLY*WY+FLZ*WZ
WSOR(I,J,K)=WSOR(I,J,K)+RHOW(I,J,K)*(WL(I,J,K,L)/DT-FLUX)
100     CONTINUE
!-----------------------------------------------------------BC
DO 210 K=KKST,KKEND
DO 210 J=JJST,JJEND
IF(IBCW.EQ.IOUT .OR. IBCW.EQ.ISYM) AW(    2,J,K)=0.
IF(IBCE.EQ.IOUT .OR. IBCE.EQ.ISYM) AE(NIM,J,K)=0.
210     CONTINUE

DO 220 K=KKST,KKEND
DO 220 I=IIST,IIEND
IF(IBCS.EQ.IOUT .OR. IBCS.EQ.ISYM) AS(I,    2,K)=0.
IF(IBCN.EQ.IOUT .OR. IBCN.EQ.ISYM) AN(I,NJM,K)=0.
220     CONTINUE
!-----------------------------------------------------------------
IF(IBOIL.EQ.0) GOTO 390
DO 300 K=KKST,KKEND
DO 300 J=JJST,JJEND
DO 300 I=IIST,IIEND
IF(T_2PH(FW(I,J,K),FW(I-1,J,K))) THEN
DWIM=(FW(I,J,K)*AMZ(I-1,J,K)-FW(I-1,J,K)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I-1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AW(I,J,K)*DWIM
ENDIF

IF(T_2PH(FW(I,J,K),FW(I+1,J,K))) THEN
DWIM=(FW(I,J,K)*AMZ(I+1,J,K)-FW(I+1,J,K)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I+1,J,K))*VFG
SS(I,J,K)=SS(I,J,K)+AE(I,J,K)*DWIM
ENDIF

IF(T_2PH(FW(I,J,K),FW(I,J-1,K))) THEN
DWIM=(FW(I,J,K)*AMZ(I,J-1,K)-FW(I,J-1,K)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I,J-1,K))*VFG
```

```
SS(I,J,K)=SS(I,J,K)+AS(I,J,K)*DWIM
ENDIF

IF(T_2PH(FW(I,J,K),FW(I,J+1,K))) THEN
DWIM=(FW(I,J,K)*AMZ(I,J+1,K)-FW(I,J+1,K)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I,J+1,K))*VFG
SS(I,J,K)=SS(I,J,K)+AN(I,J,K)*DWIM
ENDIF

IF(T_2PH(FW(I,J,K),FW(I,J,K-1))) THEN
DWIM=(FW(I,J,K)*AMZ(I,J,K-1)-FW(I,J,K-1)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I,J,K-1))*VFG
SS(I,J,K)=SS(I,J,K)+AB(I,J,K)*DWIM
ENDIF

IF(T_2PH(FW(I,J,K),FW(I,J,K+1))) THEN
DWIM=(FW(I,J,K)*AMZ(I,J,K+1)-FW(I,J,K+1)*AMZ(I,J,K))&
&       /ABS(FW(I,J,K)-FW(I,J,K+1))*VFG
SS(I,J,K)=SS(I,J,K)+AT(I,J,K)*DWIM
ENDIF
300     CONTINUE
!-------------------------------------------------------
390     CONTINUE
DO 400 K=KKST,KKEND
DO 400 J=JJST,JJEND
DO 400 I=IIST,IIEND
WSOR(I,J,K)=WSOR(I,J,K)+SS(I,J,K)
AP(I,J,K)=RHOW(I,J,K)/DT+AE(I,J,K)+AW(I,J,K)+AN(I,J,K)+AS(I,J,K)+AT(I,J,K)+AB(I,J,K)
400     APC(I,J,K)=AP(I,J,K)

IF (TASKID.NE.0) THEN
CALL
MPI_SEND(WSOR(IIST:IIEND,JJST:JJEND,KKST:KKEND),SPANX*SPANY*SPANZ,MPI_DOUBLE_PR
ECISION,0,TASKID,MPI_COMM_WORLD,IERR)
ENDIF

IF (TASKID.EQ.0) THEN
DO NT=1,26
CALL
MPI_RECV(WSOR(RSPANX(NT):RSPANXE(NT),RSPANY(NT):RSPANYE(NT),RSPANZ(NT):RSPANZ
E(NT)),SIZECC(NT),MPI_DOUBLE_PRECISION,NT,NT,MPI_COMM_WORLD,STATUS,IERR)
ENDDO
ENDIF

CALL MPI_BARRIER(MPI_COMM_WORLD,IERR)
CALL
MPI_BCAST(WSOR(1,1,1),SIZE(WSOR),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,IERR)

RETURN
END
```